Experience Report:
Peer Instruction in Introductory Computing

Beth Simon, Michael Kohanfars, Jeff Lee, Karen Tamayo
Computer Science and Engineering Department
University of California, San Diego
La Jolla, CA USA
bsimon@cs.ucsd.edu, {mkohanfa, jwl001, ktamayo}@ucsd.edu

Quintin Cutts
Department of Computing Science
University of Glasgow
Glasgow, Scotland.
quintin@dcs.gla.ac.uk

ABSTRACT
Peer Instruction (PI) is a pedagogical technique to increase engagement in lectures. Students answer a multiple choice question (MCQ) typically using clickers (hand-held remote devices with a minimum of 5 option buttons), discuss the question with their peers, and then answer the question again. In physics, PI has years of evidence of increased conceptual learning, as measured by the Force Concept Inventory (FCI) [7]. In this experience report, we describe how PI was applied in CS1 and CS1.5 courses teaching Java. We identify specifics of the standard PI model which were adopted, adapted, or discarded for use in introductory computing, describe the process involved for the instructor, give examples of the types of questions asked of students, report on students’ performance in answering these questions, reflect on the value for the instructor, and report the attitudes and opinions of the students. We conclude with observations, advice and suggested improvements.

Categories and Subject Descriptors
K.3.2 [Computer Science Education]: Introductory Programming – abstract programming concepts

General Terms
Human Factors

Keywords
CS1, peer instruction, clickers, PRS, classroom response, active learning.

1. INTRODUCTION
Peer Instruction (PI) is a well-documented method of instruction which has been shown to improve student performance on conceptual questions in physics for almost 20 years [4]. Specifically, PI challenges students to work with their deep understanding of the core concepts of the subject rather than the simpler rote application of formulae. The specific PI format is to pose students a ConcepTest (a MCQ requiring deep conceptual understanding) to answer individually using clickers, then have them discuss the question in groups, challenging each other on their answers, and after reaching a group consensus, to answer again. Often this process is accompanied by reading quizzes which students complete before class (requiring them to read material that won’t be lectured on, to make time for ConcepTests and discussion), mini-lectures before a ConcepTest covering difficult issues from the reading, and post-ConcepTest discussion led by the instructor. Student performance is shown to increase on both standardized measures (such as the FCI) – and on control comparisons of final exam questions [4].

Here we report on the process and results of applying the PI model in introductory computing courses. Our goal was to leverage the benefits the PI model has shown for engaging students in deep understanding of core concepts – but in the context of introductory computing challenges. We describe the process by which ConcepTests were developed, the kinds of questions developed, the way in which they were presented and used in class, how students fared answering these questions, and student valuation of PI assessed through survey questions. We conclude with recommended advice and suggested improvements.

2. RELATED WORK
The seminal work on PI in physics was reported in [4]. However, additional summative reports in other disciplines can be found [2]. Best practices guides are also available [12].

In the computing education context, use of a specialized web application called MessageGrid was reported in an algorithms and data structures course. [10] Pre-lecture reading assignments were given and mini-lectures (10-15 min) were followed by audience questions and discussion using MessageGrid following the PI model. Students reported the process was valuable for learning. However, that work did not evaluate the student performance on MessageGrid questions, discuss how questions were developed, or note how many questions students answered throughout the term.

Clickers are used in [9] in a short course CS1 in Java, asking questions approximately every 15 minutes. However, they did not use the PI instructional model and do not report on student responses to questions. Detailed results are provided in [3] from a one-week experiment using clickers. However, [3] differed substantially from the PI model, using clickers for a 10-15 minute comprehension test at the beginning of class, followed by group work, reported back at the end of class via use of clickers.

Cutts et al. developed a special feedback session of PI questions derived directly from issues arising classwide in a written midterm exam, in order to engage the students fully with aspects of the course demonstrated to be problematic [5].
3. SETTING
In this section we describe the courses where PI was used and the adaptations we made to the general PI model.

3.1 The Courses
We report on two introductory computing courses taught at a large US research intensive public university on the quarter system (10-week terms). The courses are a Java-based CS1 and CS1.5 (first two terms) of introductory computing intended for students with no prior programming experience. PI was used for the first time in Fall 2008. The courses reported here are from CS1 in Winter 2009 and CS1.5 in Spring 2009, i.e. the second time they were taught. We present these “second trys” as reflecting slightly polished “first attempts” at using PI in introductory computing. The same instructor (Simon) taught both courses, and 60% of the students from CS1 enrolled in CS1.5.

Table 1. Course Size and Pass Rate

<table>
<thead>
<tr>
<th>Course</th>
<th>Pass</th>
<th>D or F</th>
<th>Withdraw</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS1 (N=94)</td>
<td>86%</td>
<td>7%</td>
<td>7%</td>
</tr>
<tr>
<td>CS1.5 (N=69)</td>
<td>88%</td>
<td>3%</td>
<td>9%</td>
</tr>
</tbody>
</table>

3.2 Model Adaptation
The PI model does not simply involve having students answer questions with clickers in class. Within the PI literature (primarily in physics) there are specific implementation features of PI that, when applied, have been shown to have strong positive impacts on learning. The instructor adopted some aspects of the PI model faithfully, but also deviated from the standard model in some aspects. The standard components are itemized below with bold indicating whether they were adopted here.

- Textbook reading was assigned before each class
- Reading Quizzes (before or at the beginning of class, to encourage students to complete the reading and self-test their understanding of it) were not used due to preparation cost.
- Mini-lectures were not used. A common PI tradition is to have 10-15 minute mini-lectures prior to a set of ConcepTests. Here, the instructor felt that textbook reading was sufficient to enable students to immediately engage with PI questions.
- Clicker questions, like the physics ConcepTests, did aim to challenge deep understanding.
- The typical PI vote-discuss-vote format on clicker questions was used. Students initially had 30-90 seconds to answer; they then discussed the question with their peers, and were expected to come to a consensus before answering again.
- Discussion groups were not pre-assigned, but were ad-hoc (e.g. discuss with those people around you) due to implementation costs (lack of numbered seats).
- The result of the first vote was not always hidden from students before they entered into discussion. This seemed to promote interest, but may encourage voting for the most popular answer.
- The correct answer was indicated on the slide after discussion.

- Students were asked to provide reasonable explanations for why wrong answers were wrong, in their own words (in addition to explanations of the correct answer).
- The instructor repeatedly presented students with rationale and research results from physics showing the benefits of the PI model for learning.

4. QUESTION DEVELOPMENT
A significant question is whether the clicker questions asked were similar to physics ConcepTests, in testing deep understanding of core concepts. The differences in introductory courses in these two disciplines make this a difficult question to answer.

It is rare in the PI research literature to see analysis of questions posed in class, and therefore difficult to determine precisely the nature of an effective PI question. Physics has the widely-accepted FCI, and so it is clear to physics teachers which concepts should be addressed. There is no such inventory in computing around which to base PI questions. Faced with that lack, therefore, we describe here the process and thought by which the instructor generated questions for these courses and give some examples of the variation in types of questions asked.

PI questions were developed with the expectation that students had completed the textbook reading before class, though perhaps not thoroughly comprehended all of it. To develop a question, the instructor scanned a few pages of the text, identified a concept or programming structure that from prior experience she expected students might not thoroughly grasp, or for which common usage errors often occur – then designed a question to expose that issue, and created 3-4 distractors that would be correct given a specific misconception. In line with the PI literature, questions were written emphasizing application of concepts over, for example, simple language rule knowledge questions. Such simple questions can cause student boredom and resentment [13, 2].

![Image](346x212 to 530x351)

Figure 1. A Language Feature Conceptual Question

Obvious questions are code tracing or “select the right line(s) of code” questions (similar to those reported in [8]). A simple example might be the incorrect use of <= instead of < in a loop condition. More conceptual questions might test programming structure behavior (see Figure 1) or ask students to pick the correct diagram of the order in which a 2-D array was accessed (Figure 2).

Some questions asked students to express higher-level analysis such as explaining in English words what a code fragment does [8] or describing conditions when a particular code structure or feature should be used. In more complex scenarios, a series of
questions can be used to have students “write” code (e.g. after discussion of a recursive algorithm to perform binary search, select the line of code to use as a method header, select the line of code that correctly implements the base case, select the line of code for the recursive case where you are looking for something “earlier”, etc.).

Table 2 shows the number of questions given in each class along with an average number per 1 hour 20 minute lecture period.

Table 2. Peer Instruction Questions In Each Class

<table>
<thead>
<tr>
<th>Course</th>
<th>Total Clicker Questions</th>
<th>Questions per 80 minute lecture</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS1 Wi</td>
<td>71</td>
<td>4.4</td>
</tr>
<tr>
<td>CS1.5 Sp</td>
<td>66</td>
<td>4.1</td>
</tr>
</tbody>
</table>

5. THE IN-CLASS EXPERIENCE

Using most commercial clicker systems, the instructor can see student responses as they submit them (often as counts per answer). Whether or not to display these results on the projected screen is usually an option. In class, the instructor always checked to see how students answered on their individual (first) vote on a question. In the rare case that almost all students got it right, the instructor often just reported this to the students and briefly discussed the right answer. This may not have been effectively implemented, as it can be quite challenging, in the middle of class, to make a good judgment on whether a topic has already been mastered by students. In the second running of each course, the instructor was much more likely to simply always have students discuss and revote, being more confident in the quality of the questions or the general value of students discussing answers and putting their explanations into their own words.

Students required a bit of prompting the first class or two – they are not used to talking being a necessary part of the class. Several requests by the instructor of “It should be louder -- talk! Explain why you think the right answer is right and why the wrong answers are wrong” for the most part got an anecdotal 90% or more participation. This last sentence was repeated often. Students were frequently reminded that being able to put into words exactly why wrong answers were wrong would help them cement critical issues in their minds, hopefully helping them identify mistakes while doing programming assignments or answering exam questions.

The amount of time spent on each question, for individual or group voting or discussion, varied. In general, 30-60 seconds were given for the individual vote, though often the instructor had read the question to the students first. Discussion time varied (but was around 3-5 minutes), with the instructor roaming around the room listening in and offering to answer questions. This allowed the instructor to get a feel for when useful discussion was wrapping up. The final vote was usually very quick – less than 20 seconds. Students always want to see the result of the final vote. Discussion always followed. Usually the instructor would confirm that the most popular answer was, in fact, correct (and indicate it with Tablet PC ink on the slide). Often she then (using the vote results either of the final vote or the first vote) asked students about a specific, popular wrong answer (or 2 or 3) – asking a student in the audience to provide an explanation of why that answer would be tempting. Since students worked in groups discussing these issues, they seem prepared and more confident in speaking these explanations in front of the rest of the class. Additionally, perhaps because it was an instructor-selected “mistake” or because they know other students voted for that answer, there seems to be less stigma with explaining the reasoning that made that answer seem plausible (especially when they can go on to say how they figured out it was wrong).

6. PERFORMANCE ON PI QUESTIONS

How did students do on the questions, and did they improve in the “after discussion” vote? The instructor didn’t have any firm goals with regards to a target correctness level either for the before or after discussion vote, other than to make questions challenging enough to keep students engaged and to give them impetus to discuss with their peers. However, Mazur reports “the number of students who give the correct answer to a ConceptTest increases substantially, as long as the initial percentage of correct answers to a ConceptTest is between 35% and 70%. (We find that the improvement is largest when the initial percentage of correct answers is around 50%)”[4].

Table 3. PI Question Counts by Initial Vote Correctness

<table>
<thead>
<tr>
<th>Course</th>
<th>&lt;35%</th>
<th>[35,70]</th>
<th>&gt;70%</th>
<th>Highest</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS1</td>
<td>21%</td>
<td>68%</td>
<td>11%</td>
<td>86%</td>
</tr>
<tr>
<td>CS1.5</td>
<td>26%</td>
<td>70%</td>
<td>4%</td>
<td>76%</td>
</tr>
</tbody>
</table>

For these classes 68%-70% of the questions fell within the initial vote correctness recommended by Mazur. Of the rest of the questions, the majority were quite hard for students and no question was answered correctly by more than 86% of the class.

Both CS1 and CS1.5 classes performed consistently with the average correctness in the initial vote to be 47% and 44% and the after discussion vote to be 68% and 63% respectively. Figure 3 shows the correctness on a per question basis for CS1 only (the graph for CS1.5 shows similar characteristics). Each bar along the X-axis reports one question, and questions are sorted from lowest to highest correctness by initial vote (before discussion). The light color bars show correctness level before discussion, the dark “tops” indicate correctness level in the group vote (except when the bar goes negative, that highlights when more people got the question wrong after discussion).
Although one interpretation of this behavior is that only ~20% of students improve via peer discussion, that ignores the students who answered the question correctly first time who answer correctly in the after-discussion vote. We calculated normalized gain, NG, using the correctness, C, for each vote, as (A: after discussion, B: before):

\[
\begin{align*}
\text{If } C_A > C_B : & \quad NG = (C_A - C_B) / (1-C_B) \\
\text{If } C_A < C_B : & \quad NG = (C_A - C_B) / (C_B)
\end{align*}
\]

The normalized gain for CS1 and CS1.5 was 41% and 35% respectively. Mazur [4] reports normalized gain in one class of 43% and reports a genetics class show 33%[11]. However, these numbers were calculated from class averages, not per question (as ours were). Interestingly, our numbers do not concur with Mazur’s report that the gain is largest when the initial percentage of correct answers is around 50%. We find slight increases in normalized gain as the proportion of initially-correct answers for a question increases. Why this is the case, and greater detail on student performance analysis, is a subject for future work.

### 7. STUDENT SURVEYS

Students answered questions on the use of PI as part of an end of term survey, which was required for a point in the class. Since many of the students in CS1 are also represented in CS1.5 (60% went on to take it) similar results on their perceptions would be expected. In this section, we also report survey results from an earlier CS1.5 course (in Winter 2009) which has a different set of students in it for a better sense of student perception. These numbers are reported in parentheses in Tables 4 and 5.

Overall, students responded positively to PI in CS1 and CS1.5. Table 4 shows that over 60% of students would recommend another instructor to use it; Table 5 shows that 78-87% of them agree that PI was valuable for their learning. We also asked some questions to get students’ self-reporting on the fidelity with which they engaged in the PI process. Although the classroom seemed amazingly animated and engaged to the instructor, we find out from students that 9-17% of them rarely discussed and 30-33% only sometimes discussed, whilst the remainder always did so.

Based on instructor suspicion, we asked whether students actually read the textbook before class and confirmed that only a minority of the class (20-38%) read most or all of the time. Additionally, of the people who didn’t always read before class, about half of them felt this had had a negative impact on their ability to learn in class. These results feed into our suggested improvements.

### 8. RECOMMENDATIONS AND DISCUSSION

From the reported data we can get a picture of how the PI pedagogy was adapted and adopted in two introductory computing courses. The instructor developed questions which were, by the numbers, challenging for students, but also engendered discussion and learning (based on increased correctness after group discussion). Students, on a survey, reported to finding PI valuable for their learning.

However, the students weren’t the only ones who felt they learned more by using PI. Here we summarize some of the value the instructor felt in using PI:

- **Preparing Questions:** It can be very valuable to review the textbook, looking at it with a critical eye as to how a student would read and likely not quite understand certain concepts. Coming up with possible distractors can be both frustrating and illuminating.
- **In Class:** It was easy to figure out how much time to spend in discussion of a question, by looking at the correctness of votes both in the solo and group vote. Sometimes the instructor was surprised at both how well or how poorly students did on specific questions.
- **Engaging Students in Post-Voting Class Discussion:** Since students did vote individually, and often wrongly, it is very easy to say “many of us thought that C was correct originally, why is that reasonable? Why might you think it is C?” This seemed to lead students to be comfortable sharing their own explanations of misconceptions.
- **After Class:** It was very easy to create challenging exam questions: look for questions with low group vote correctness rates, and make students aware of this method of preparing exam questions. We are in the process of analyzing the data to determine if this is effective.

Based on this experience report, we offer up a few recommendations and mention a few concerns that will be addressing in future uses of PI.

- **Getting Started:** Look for an advisory document like [1, 12] to make sure that you invest time wisely in creating the right kinds of questions (hard, but meaningful ones) and put in the

---

**Table 4. Would you recommend that other instructors in computing courses use clickers with discussion?**

<table>
<thead>
<tr>
<th>Course</th>
<th>Agree</th>
<th>Neutral</th>
<th>Disagree</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS1, N=87</td>
<td>62%</td>
<td>28%</td>
<td>6%</td>
</tr>
<tr>
<td>CS1.5, N=60 (N=79)</td>
<td>60% (65%)</td>
<td>35% (25%)</td>
<td>3% (4%)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Table 5. Clickers with discussion is valuable for my learning.**

<table>
<thead>
<tr>
<th>Course</th>
<th>Agree</th>
<th>Neutral</th>
<th>Disagree</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS1</td>
<td>78%</td>
<td>8%</td>
<td>14%</td>
</tr>
<tr>
<td>CS1.5</td>
<td>87% (81%)</td>
<td>7% (11%)</td>
<td>7% (8%)</td>
</tr>
</tbody>
</table>
right motivational structures (points for students) to ensure the students are properly prepared.

- Promoting Student Discussion: At the beginning of discussion say “discuss with those around you. Don’t just figure out the right answer. Explain why the right answer is right, put into words why the wrong answers are wrong”. Focus students on evaluating all the answers, not just finding the right one.

- Engaging Students in the Process: It is important to inform students about your choice of pedagogy, and explain to them the benefits of using it. Showing graphs from research papers (even in other fields) can be helpful, to provide evidence to students that this non-traditional form of instruction is designed to help them succeed.

Issues which we discovered in the use of PI which we plan on addressing in future use include the following:

- Getting Students to Read: We’ll implement the suggested PI practice of simple reading quizzes for credit (given online, before class) so that students have an incentive to read the textbook before class.

- Showing Voting: We’ll switch to a default of not showing solo vote results – to keep students focused on evaluating all the answers rather than simply switching to the most popular.

- Improve Questions: We’ll review and possibly modify questions with low initial vote correctness, especially if they have a low normalized gain.

- Better Student Discussion: We’ll assign discussion groups, rather than letting students self-form into groups. We have particular concerns about under-represented groups feeling uncomfortable. We’ll choose groups of size 4 to allow for missed classes and attrition.

The PI model is not commonly used in CS. Anecdotally, we know that students struggle to successfully apply introductory programming concepts, exactly the area in physics where PI has been so successful. Yet in CS, traditional lecture models appear to be common with instructors explaining and perhaps working examples for students in an attempt to promote learning and understanding. We hope that this report will give instructors some understanding of how PI may be adapted to computing.

Inevitably, instructors will also value additional evidence of the positive impact of PI on student learning in CS. We have shown here that learning “by peer discussion” (as measured by normalized gain from individual vote to group vote) does achieve similar results to those reported in physics and genetics. Crucially, it should be noted that the 44% “in-class” normalized gain reported in [4] was only preliminary evidence of learning – the final normalized gain in that class, based on the FCI, was 74%. Draper [6] and Smith et al [11] both argue that this difference between immediate and final learning gain is to be expected when using PI. An analysis of our students’ voting behaviour against their final result is underway; in future work we hope to further elucidate the impact of PI on CS learning. The lack of a validated instrument such as the FCI will add challenges.

9. CONCLUSIONS
This experience report describes the process and results of using Peer Instruction in two introductory computing courses. The instructor developed a set of varied, challenging multiple choice questions based on difficulties and misconceptions students might be expected to have from reading the textbook. Lecture was replaced with a process of asking a series of questions (4-5 in 80 minutes) in the PI vote, discuss, revote model. Finally, the instructor led discussion of the question, engaging students. Students’ initial correctness in answering ranged from 15%-86% with an average of 44-47%. Average correctness after discussion was 63-68% with a normalized gain of 35-41%. Students were generally very positive towards PI with 78-87% reporting it was valuable for their learning. The instructor reported value in being able to easily gauge explanation depth after asking questions and in finding key issues students had not yet mastered for developing quizzes and exams. We provide a set of recommendations and issues to address for future use of PI in introductory computing.

10. REFERENCES