Peer Instruction in the CS Classroom: A Hands-On Introduction

Daniel Zingaro
Department of Computer Science
University of Toronto
Toronto, ON, Canada
daniel.zingaro@utoronto.ca
Leo Porter, Beth Simon
Computer Science and Engineering Department
University of California, San Diego
La Jolla, CA, USA
leporter@cs.ucsd.edu, bsimon@cs.ucsd.edu
John Glick
Department of Mathematics and Computer Science
University of San Diego
San Diego, CA, USA
glick@sandiego.edu

Contents

1 PI Process

Here is a quick description of the Peer Instruction (PI) procedure [14, 3].

Pre-lecture. Acknowledging that PI very likely reduces the amount of material we can "cover" in lecture, many instructors require the completion of textbook reading and an associated reading quiz prior to each class. In what is referred to as just-in-time teaching (JiTT), instructors can use the reading quiz submissions to structure lecture around material found problematic by students. This maximizes the utility of the limited available lecture time, and helps us target actual student misconceptions rather than our perceptions of those misconceptions.

PI Proper. In lecture, we run a number of iterations of the PI procedure; we have been able to use 2-4 ConcepTests per 50-minute lecture period. Each iteration begins with the posing of a ConcepTest: a multiple-choice question (MCQ) targeting the recall, analysis and elaboration of key concepts and material. 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). Next, we display the histogram of results: this helps students see that they are not alone in their confusion, and indicates some of the misconceptions possessed by students. We use this information to inform subsequent class-wide discussion, in which we discuss each answer and argue why it is correct or incorrect. The focus in this discussion is not to "answer the question", but to engage in decision-making and perspective-building that shape how students approach the subject.

POST-LECTURE. We have argued [14, 5] that it is important to permeate PI culture through all aspects of course delivery. For example, tutorials and labs should give students further opportunity for student-initiated dialogue and discussion. TAs should be given information from lecture to enable them to address remaining student misconceptions (uncovered through ConcepTests) in small-group settings.

2 How do we Know PI Works?

There's lots of evidence! Here's some:

- Students consistently laud the benefits and effects of PI on their learning and enjoyment of lectures. For example, surveys in [10] found that 96% of students agreed that clickers and peer discussion helped them learn and understand the material, 96% agree that the feedback provided by clickers is helpful, and 76% agree that posting questions about pre-lecture reading helps them identify difficulties.
- PI-in-CS research consistently finds significant normalized gain (NG) between the solo vote and corresponding group vote. NG is defined as the proportion of students who answered incorrectly in the solo vote but correctly in the group vote. We have measured 41% NG in a CS1 offering, 35% NG in a CS1.5 offering [12], and 29% in a remedial CS1 course for students attempting the course for the second time [14].
- In physics, researchers have used a pre/post-design with the FCI to measure NG. In one case, NG was found to double in a calculus-based physics course upon the introduction of PI [3].
- . . . but maybe NG does not indicate learning. Maybe it results from students copying answers from neighbors they perceive to be more knowledgeable? To test this, Smith et al. [13] followed 16 PI iterations with an isomorphic question that students had to answer individually. That is, these PI iterations now consisted of three questions: Q1 (the solo vote), Q1ad (the group vote), and Q2 (a solo vote on a new, similar question). Two important findings: (1) average percentage correct on Q2 is significantly higher than Q1 and Q1ad, and (2) looking only at students who answered Q1 incorrectly but Q1ad correctly, 77% answered Q2 correctly! Apparently, these 77% of students learned something, because they were able to apply knowledge to a slightly-changed context. One more surprising finding: if we look only at students who got both Q1 and Q1ad wrong, 44% still managed to answer Q2 correctly. Next, looking at the difficulty of Q1, we find that Q2 correctness is significantly higher than Q1's at each difficulty level, and Q2 correctness is significantly higher than even Q1ad on the most difficult questions.
- In another study, some questions on the final exam mirrored ConcepTests from lecture, but used new physical contexts [4]. The primary finding is that the correctness on the final exam question is comparable to the correctness on the ConcepTest after discussion, and far higher than correctness on the solo vote.
- PI was invented at Harvard, but it also works at liberal arts colleges, even when students' background knowledge is limited. Lasry et al. [9] looked at the results of using PI for the first time in physics, comparing with the first PI course taught at Harvard. Lasry et al. also had a control class that enrolled in a traditional offering of the course. PI students did better on the final exam; but the most interesting findings have to do with background knowledge. Students were split into low-and high-background groups, based on their performance on the FCI at the start of the course. High-background PI students outperform high-background traditional students, and low-background PI students outperform low-

background traditional students. But, low-background PI students also outperform high-background traditional students: it seems we can use PI with widely varying levels of incoming knowledge.

- There is evidence that PI is equitable across genders. Reay et al. [11] used a PI-informed voting pedagogy to teach three introductory quarters of a physics course, and also had a traditional control class. Again, the voting section outperformed the traditional section on a Concept Inventory for Electricity and Magnetism, and on common conceptual questions on midterms and a final exam. Critically, males outperformed females in the traditional offering, but they performed equally in the voting section. Similar results can be found in [4]: males outperform females on the FCI following a traditional offering, but perform equally in a PI offering.
- How about us: do instructors find PI valuable? Apparently: 93% of instructors surveyed about their use of PI indicate that they had a positive experience, and 79% of instructors definitely plan to use PI again [4].

3 Generating Effective MCQs

Generating effective MCQs for PI is not easy, especially in light of a lack of freely-available PI ConcepTests in CS. In this section, we summarize some research that gives concrete advice and avenues for exploration when developing questions. Perhaps most importantly: be clear on the goals and purpose of a question before using it [1]!

3.1 Advice from Beatty et al.

Beatty et al. [1] provide a useful framework for question design. (They developed it for physics, but we think it is more widely applicable.)

Every question should have three goals:

- Content goal: what is the course "material" that we want the question to probe?
- Process/cognitive goal: what "habits of mind" are required to answer the question? e.g. generate multiple solutions, compare and contrast, categorize and classify, think about thinking, etc.
- Metacognitive goal: what beliefs about teaching and learning our subject do we want the question to reinforce? For example: what is CS about? What role does cooperation play in the discipline? Is confusion and resolution necessary?

Beatty et al. give many tactics for designing questions that meet their desired goals. Here are some examples:

COMPARE AND CONTRAST. This can be used to direct attention to the differences between two situations. One way to accomplish this is to ask the same query about several related questions; Figure 1 provides a two-ConcepTest sequence designed to do this.

EXTEND THE CONTEXT. Another "directing-attention" tactic: ask a familiar question in a new context. For example, we can ask about finding the length of a string early in the course, and then ask again once recursion has been introduced (Figure 2).

OOPS-GO-BACK. This is another two-question tactic, both of which are presented in sequence without instructor intervention. The first question is a trap meant to cause students to make a mistake, and the second question is meant to alert students to the mistake they just made.

CATEGORIZE AND CLASSIFY. In this one, we ask students to identify a subset meeting a given criterion (Figure 3).

1. The following code template counts the total number of asterisks reachable from location `(x, y)`. Select the code to use in place of the comment.

```
int blobSize (char grid[][COLS], int x, int y) {
  int left, right, up, down;
  if (grid[x][y] == ' ')
    return 0;
  else {
    grid[x][y] = ' ';
    left = blobSize (grid, x, y-1);
    right = blobSize (grid, x, y + 1);
    up = blobSize (grid, x - 1, y);
    down = blobSize (grid, x + 1, y);
    return ... // fill in the code
  }
}
```

- A. `1 + left + right + up + down;`
- B. `left + right + up + down;`
- C. `4 + right + left + up + down;`
- D. `right + left + up + down;`

2. We've commented out some code. Is the solution still correct?

```
int blobSize (char grid[][COLS], int x, int y) {
  int left, right, up, down;
  if (grid[x][y] == ' ')
    return 0;
  else {
    //grid[x][y] = ' ';
    left = blobSize (grid, x-1, y);
    right = blobSize (grid, x+1, y);
    up = blobSize (grid, x, y-1);
    down = blobSize (grid, x, y+1);
    return 1 + left + right + up + down;
  }
}
```

- A. No, because the code could loop indefinitely
- B. No, because the grid will now be modified when the function finishes
- C. No, but I don't know why
- D. Yes

Figure 1: Compare and Contrast

CONSTRAIN THE SOLUTION. Require that students use a particular approach or refrain from using a specific approach.

REVEAL A BETTER WAY. Intentionally present a question that students will try to solve using a tedious or error-prone technique. Group-and class-discussion then reveals the more elegant route that could be used.

MULTIPLE DEFENSIBLE ANSWERS. Use qualitative questions where there is no one best answer: this is very useful for generating productive discussion. Similar discussion-producing tactics include introducing deliberate ambiguity and trapping unjustified assumptions.

Which of the following correctly finds the length of string `s`?

- A.

```
int findLen1 (char *s) {
  if (*s == '\0')
    return 0;
  else
    return 1 + findLen1 (s+1);
}
```

- B.

```
int findLen2 (char *s) {
  if (*s == '\0')
    return 1;
  else
    return 1 + findLen2 (s+1);
}
```

- C.

```
int findLen3 (char *s) {
  if (*s == '\0')
    return 0;
  else
    return findLen3 (s+1) + 1;
}
```

- D. Two of the above are correct
- E. All of the above are correct

Figure 2: Extend the Context

Here are some if-statements, with placeholders for their bodies.

(I) `if (2 < 3) S1;`

(II) `if (2) S2;`

(III) `if (0 == 0) S3;`

(IV) `if (0) S4;`

Which statements will be executed?

- A. (I)
- B. (I) and (II)
- C. (I) and (III)
- D. (I), (II) and (III)
- E. (II) and (IV)

Figure 3: Categorize and Classify

3.2 Advice from other Literature

Here are some more ideas for generating MCQs, straight from the general PI literature and specific PI-in-CS literature:

PARSON PUZZLES. Parson puzzles are sometimes used on CS exams in order to target an intermediate skill between code-reading and code-generation. These exercises ask students to reorder given code into a configuration that solves a specified problem [6]. We have experimented with Parson Puzzles in PI [14] and have witnessed increased discussion and focus. The questions are challenging, yet manageable because "the answer is on the slide". See Figure 4 for an example.

MULTIPLE QUESTIONS PER CONCEPT. When we use only one MCQ per concept, we are not giving students the opportunity to generalize their knowledge across contexts. Reay et al. [11] sug-

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 4: Using a Parson Puzzle as a ConcepTest.

gest using easy-difficult-difficult and "rapid-fire" question sequences: the former to assess just how deep the students' knowledge runs; the latter to give students targeted practice with a new concept.

4 Final Tips

ConcepTests should [4, 1, 2]:

- Contain at most five response choices
- Focus on a single important concept
- Require thought, not rote application of a process
- Provide plausible distractors that reveal likely student difficulties
- Be unambiguous
- Be challenging (e.g. 35-70% correctness on the solo vote)
- Have a content (concept), cognitive (process), and metacognitive goal
- Contain only information that is essential to the pedagogic purpose

Carefully consider whether to use low-stakes (all answers, including incorrect answers, get the same points), and high-stakes (correct answers get more points) grading:

- High-stakes grading encourages students to respond with the preference of another student [8]
- Students in low-stakes grading are more likely to submit different answers than peers [7]
- High-stakes grading is associated with discussions dominated by one person (usually the most knowledgeable person) [7]
- Regardless, make clicker scores available on a regular basis to reduce student anxiety [2]

5 Available ConcepTest Banks

Here are some pointers to existing PI ConcepTest banks. (Of course, we hope that you will contribute your own questions soon!)

- ConcepTests and reading quizzes for a C-based CS1 are available at [15].

References

[1] BEATTY, I. D., GERACE, W. J., LEONARD, W. J., AND DUFRESNE, R. J. Designing effective questions for classroom response system teaching. *American Journal of Physics 74* (2006), 31–39.

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

[3] CROUCH, C. H., AND MAZUR, E. Peer instruction: Ten years of experience and results. *American Journal of Physics 69* (2001), 970–977.

[4] CROUCH, C. H., WATKINS, J., FAGEN, A. P., AND MAZUR, E. Peer instruction: Engaging students one-on-one, all at once. In *Research-Based Reform of University Physics*, E. F. Redish and P. J. Cooney, Eds. American Association of Physics Teachers, 2007.

[5] CUTTS, Q., KENNEDY, G., MITCHELL, C., AND DRAPER, S. 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)., 2004.

[6] DENNY, P., LUXTON-REILLY, A., AND SIMON, B. Evaluating a new exam question: Parsons problems. In *ICER '08: Proceedings of the Fourth international Workshop on Computing Education Research* (New York, NY, USA, 2008), ACM, pp. 113–124.

[7] JAMES, M. C. The effect of grading incentive on student discourse in peer instruction. *American Journal of Physics 74* (2006), 689–691.

[8] JAMES, M. C., AND WILLOUGHBY, S. Listening to student conversations during clicker questions: What you have not heard might surprise you! *American Journal of Physics 79* (2011), 123–132.

[9] LASRY, N., MAZUR, E., AND WATKINS, J. Peer instruction: From harvard to the two-year college. *American Journal of Physics 76* (2008), 1066–1069.

[10] PARGAS, R. P., AND SHAH, D. M. Things are clicking in computer science courses. In *SIGCSE '06: Proceedings of the 37th SIGCSE technical symposium on Computer science education* (New York, NY, USA, 2006), ACM, pp. 474–478.

[11] REAY, N. W., LI, P., AND BAO, L. Testing a new voting machine question methodology. *American Journal of Physics 76* (2008), 171–178.

[12] SIMON, B., KOHANFARS, M., LEE, J., TAMAYO, K., AND CUTTS, Q. Experience report: Peer instruction in introductory computing. In *SIGCSE '10: Proceedings of the 41st SIGCSE technical symposium on Computer science education* (New York, NY, USA, 2010), ACM, pp. 341–345.

[13] SMITH, M. K., WOOD, W. B., ADAMS, W. K., WIEMAN, C., KNIGHT, J. K., GUILD, N., AND SU, T. T. Why peer discussion improves student performance on in-class concept questions. *Science 323* (2009), 122–124.

[14] ZINGARO, D. Experience report: Peer instruction in remedial computer science. In *Ed-Media 2010: Proceedings of the 22nd World Conference on Educational Multimedia, Hypermedia & Telecommunications* (2010), AACE.

[15] ZINGARO, D. Pi-cs resource page. `www.danielzingaro.com/pics.php`, 2010.