Peer Instruction in Computing: What, Why, How?

Daniel Zingaro

Abstract: Peer Instruction (PI) transforms a standard, passive lecture into an opportunity for students to answer questions individually and in groups. What is PI? Does it work in computing courses? Research evidence aside, how does one go about using PI in practice? In this paper, I describe the PI process and summarize the research that has been done on PI in CS. I then offer a personal account of how I use PI and what I have learned, including thoughts on transitioning courses from lecture-style to PI-style.

1   What is PI?

Peer Instruction (PI) is a pedagogical technique developed in physics that we have co-opted for computing. Physics educators realized that standard lectures are ineffective for teaching core concepts and addressing misconceptions (Crouch, Watkins, Fagen, & Mazur 2007). Paper after paper in physics education journals demonstrate that on any conceivable measure, PI-students outperform lecture-students: on final exams and standard concept inventories (Crouch et al. 2007), on gender-equal learning gains and lecture attendance (Reay, Li, & Bao 2008), on student attrition and problem-solving ability (Lasry, Mazur, & Watkins 2008).

Algorithmically, PI works as follows. Prior to each lecture, students are expected to read a section of the textbook and complete an associated reading quiz. We then replace our lecture with several iterations of the following four-step loop body:

PRESAGE. Each PI iteration begins with a mini-lecture: a couple of slides that act as a refresher for what students read before class. The mini-lecture lasts at most a few minutes.

ENGAGE. We next pose a ConcepTest (Crouch et al. 2007) — a multiple-choice question designed both to focus attention on and raise awareness of the key course concept from the mini-lecture (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).

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. I recommend (Caldwell 2007) for an introduction to the logistics and uses of clickers, though keep in mind that clickers are not the same as PI (clickers are a technology, PI is a pedagogy).

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 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.

2   Why PI in Computing?

A paper at SIGCSE 2010 (Simon et al. 2010) mobilized several investigations into the applicability and effectiveness of PI in CS. This original paper was primarily interested in the following question: do more students answer correctly after a group discussion compared to before the group discussion? To quantify such gains, the authors computed normalized gain (NG) on each question. NG captures the proportion of students who answer incorrectly in the solo vote but correctly

Consider the following code.

```
void recur (int i) {
  if (i <= 1) {
    printf ("hi\n");
    return;
  }
  recur (i - 1);
  recur (i - 2);
}
```

How many times is `hi` printed by the call `recur (3)`? (D)

- A. 0
- B. 1
- C. 2
- D. 3
- E. 4

Consider the following code.

```
void recur (int i) {
  if (i <= 2) {
    printf ("hi\n");
    return;
  }
  recur (i - 1);
  recur (i - 2);
}
```

How many times is `hi` printed by the call `recur (3)`? (C)

- A. 0
- B. 1
- C. 2
- D. 3
- E. 4

Figure 1: Two isomorphic questions on recursion.

in the group vote. For example, if 60% of students answer correctly in the solo vote and 80% answer correctly in the group vote, the NG for the question is 50% (i.e. 50% of "potential learners" demonstrated new understanding). The authors found an NG of 41% in a CS1 course and 35% in a CS1.5 course.

There are three concerns with this study's specific PI implementation and the NG metric in general. First, pre-class reading quizzes and mini-lectures — two staples of PI (Crouch et al. 2007) — were not used. Second, NG does not delineate students who "actually learned" from those that flat-out copied from neighbors. In addition, NG ignores what may have been learned by students who answer correctly in the individual vote. Third, solo-to-group vote spans at most a few minutes: we are left speculating on the impacts of PI on the final exam and beyond.

Zingaro (2010a) used "textbook PI" in a remedial CS1 course and found an NG of 29%. Whether this slightly lower NG resulted from the enrollment of weaker students, or the smaller class size, or some other unknown factor, it is compelling to suggest the apparent fact that almost one-third of the class learned something in the two-minute peer discussions.

Apparent, indeed. Smith et al. (2009) sought to separate peer-mirroring from genuine learning through the use of isomorphic questions in a biology course. A pair of isomorphic questions test the same concepts, but are dressed differently (see Figure 1 for an example). (In the examples in this paper, I have added the correct answer following the question text.) Smith et al. solicited an individual vote on a first question (the solo vote), had students discuss and re-vote on that question (the group vote), then had students individually answer a new, isomorphic question (the isomorphic vote). Does the correct-ness rate on the isomorphic vote drop back down to the level of the solo vote? Fortunately, no: of the students who answer incorrectly in the solo vote and correctly in the group vote (whom we will refer to as the potential learners), 77% answer correctly in the isomorphic vote. And, fortunately again, this finding transfers from biology to CS, according to a study of upper-level Architecture and Theory of Computation courses (Porter, Bailey-Lee, Simon, & Zingaro 2011). In Architecture and Theory of Computation, respectively, 68% and 53% of potential learners benefited from group discussion. If we take those students who answer correctly in both the solo and group votes as a control group, we can compare the performance of the potential learners against those who are expected to answer correctly in the isomorphic vote. Doing so, it is reported that potential learners are 89% as likely to answer correctly in the isomorphic vote as compared to the control group.

As of this writing, we have no direct research into the effect of PI on longer-term learning gains in CS. (Care to help us?) Yet, surveys consistently find that students and instructors **think** that students are gaining appreciable value from PI. A survey reported in the Zingaro (2010a)study found, for example, that 100% of students believed that individually thinking about and responding to a PI question helped them learn the course material. In addition, 100% agreed that group discussion was helpful, and 94% strongly agreed that PI should be used in other courses. Other surveys corroborate these findings (Pargas & Shah 2006; Simon et al. 2010). A unique reflection of four instructors' use of PI (Porter, Bailey Lee, Simon, Cutts, & Zingaro 2011) suggests other observed benefits, including a focus on teamwork and communication; flexible, demand-driven lectures; and opportunities for improving proof-reasoning abilities.

3 How I use PI

This section is decidedly personal opinion. It contains the advice I would offer if asked to provide a how-to on PI. I am far closer to a PI beginner than a PI expert, having used PI in two (or two and a half) different courses so far. I hope that this positions me in such a way as to have learned something useful to share with you, but from the perspective of clearly remembering what was involved in "making the switch". My PI courses include a C-based CS1, a Python-based CS1, and an Operating Systems course (the latter of which I am PI'ing incrementally, as described below).

3.1 Convince the Students

In my first lecture, I include rationale for why I am using PI. I discuss how PI can strengthen students' conceptual understanding and help me adapt lectures to pinpoint evoked difficulties. I tell students that they know how to read; instead of echoing the textbook, I will use lecture for more defendable purposes. I throw up some graphs from (Zingaro 2010a; Simon et al. 2010) and do a funny little intro PI question to introduce the process and get students comfortable with discussing and voting ("For how many hours will you study on Spring Break?").

To encourage participation, I make reading quizzes and PI involvement worth 5-10% of the course grade. Marks are awarded for participation rather than correctness, so that students can submit "their answer", rather than feeling compelled to copy in order to get marks. I tell students that I will include some PI questions on the midterm and final exam, which I hope conveys my feeling for the importance of PI questions.

3.2 Trim Knowledge from Lecture

Reading quizzes are intended to give students the required knowledge on new concepts. Students will not necessarily fully understand each concept, but they will have seen it prior to lecture. In my opinion, the major leap as a PI lecturer is to proceed under the assumption that students have done the reading and have learned at least the basics. In practice, this means cutting back on including rote textbook knowledge from slides.

I will begin here a discussion of one particular lecture: a first C-based CS1 lecture on arrays. The reading quiz has students read relevant textbook sections and answer two knowledge-level questions about arrays:

- If a float is four bytes, how much memory is occupied by `float f[6]`?
- Where is the bug in the following code? (The code has an off-by-one error on array indices.)

Students submit their reading quizzes electronically. Before class, I briefly read the responses and create a "Feedback from Reading Quiz" slide. This slide addresses core misconceptions from the reading quiz, and reassures students that I read their responses before every class. If I see students wavering on reading quiz completion or response quality, I send them an email reminding them to take the exercises seriously. Prior to this arrays lecture, I notice some students confused about array initializers (is `{}` a valid initializer?), so I address this before my "planned" content. One student asked poignantly "what is the purpose of arrays?" so I give a "what if you have one thousand students in a class . . ." argument. If you're not preparing last-minute for PI lectures, or "winging it" during class, your lectures are not sufficiently targeted.

3.3 Mini-Lectures and ConcepTests

I design my first ConcepTest to solidify the understanding that each element of an array is like a regular variable. The question asks for the value of an array element after it is assigned a value dependent on other array elements. As preparation, I spend five minutes on two slides constituting a mini-lecture. The first slide gives array-related terminology, shows the array-creation and array-access syntax, and reminds students that array elements are not initialized by default. The second slide talks about ways of avoiding undefined behavior, and states (without explanation or example) that array variables are like any other variables. I'm moving quickly. Do they understand the core concepts? The ConcepTest will let me know. I acquire the votes for the solo vote, then play a loud auditorium noise in the background and ask them to discuss in their groups. (There are about 35 students in my class, sufficiently small that no one wants to be the first to break the silence.) 92% of students answer correctly in the solo vote; 96% answer correctly in the group vote. I'm pleased and move on, though I do first ask students to help me trace the ConcepTest code, just in case.

Can students work with arrays and for-loops? Students have seen for-loops previously, so I put up just one slide showing the syntax for looping from left to right through each element of an array. Then, a ConcepTest (see Figure 2). 42% of students answer correctly in the solo vote; 54% answer correctly in the group vote. Here is where I stop to slow down! I may generate another for-loop question on-the-fly, carefully trace the code (which is what I do in this case), or have students re-discuss the question in light of another mini-lecture (since, after all, I have some evidence that the group discussion helped).

Here is the outline of a program that reads `N` numbers from the user. What code should replace `//Loop` if we want to print those numbers in reverse? (E)

```
int a[N], i;
for (i = 0; i < N; i++)
  scanf("%d", &a[i]);
//Loop
```

- A.
  ```
  for (i = N; i >= 0; i--)
    printf(" %d", a[i]);
  ```
- B.
  ```
  for (i = N - 1; i >= 0; i--)
    printf(" %d", a[i]);
  ```
- C.
  ```
  for (i = N; i > 0; i--)
    printf(" %d", a[i-1]);
  ```
- D. All of the above
- E. B and C only

Figure 2: ConcepTest testing students understanding of arrays with for-loops.

In all, the 50-minute class contains three PI iterations. (I have tried to do four in several different courses, but the fourth PI iteration inevitably feels rushed.) After class, I look in more depth at solo votes, group votes, and NGs. I care about short-term difficulties (which I can address through a "review" ConcepTest in our next class, or by preparing a handout with misconceptions and exercises), but also about overall effectiveness of my ConcepTests (are they too easy? Too difficult? Should I slow down?).

3.4 Using and Creating ConcepTests

A growing list of ConcepTests is available at (Zingaro 2010b). Even if you are teaching a course for which we do not have available ConcepTests, these should give a flavour of the types of comprehension or application questions that we like to ask.

Yet, it is likely that you will be the first to use PI in a new area of CS (compilers, databases, etc.). And in courses where ConcepTests are already available, you'll likely be making your own as you adapt existing materials for your specific context. Here are some tips for using ConcepTests:

- Before using a ConcepTest, make sure you understand its design logic (Beatty et al. 2006). The three critical questions are: what is the content knowledge being tested? (This is a single concept, such as indefinite loops or conditionals.) What cognitive mechanics are required? (For example, does the question require making comparisons, categorizing, or predicting?) What does it say about CS as a subject in general? (For example, is CS about coding or problem-solving? Is teamwork important in CS?)
- Avoid knowledge-level questions; confine those to reading quizzes. One incentive for students to take reading quizzes seriously emerges when they understand that lecture will **not** be reviewing that material. PI lectures are at a higher cognitive level than standard lectures, because we assume more of our students in terms of preparation and willingness to engage and discuss.
- I frequently draw inspiration from a typology of questions deemed useful in physics (Beatty et al. 2006). Suggestions are simple but effective and widely applicable. You can create compare-and-contrast questions that draw attention to the difference between two situations ("what happens if I create a `do` loop rather than a `while` loop?"). You can ask familiar-looking questions in new contexts ("we've seen an iterative way to calculate string-length; which of the following is a recursive way to find the string-length?"). A third technique (called oops-go-back) is to present a question that tricks students into making a mistake (based on likely misconceptions). Skip the "age" step, and follow with a second question that illuminates their mistake for an aha-moment. See Figure 3 for an example: expanding the loop

How many processes (not counting the original parent) are created by this code? (D)	How many processes (not counting the original parent) are created by this code? (D)

```
int i;
for (i = 0; i < 3; i++)
  fork();
```

```
fork();
fork();
fork();
```

- A. 3
- B. An infinite number
- C. 9
- D. 7

- A. 3
- B. An infinite number
- C. 9
- D. 7

Figure 3: An oops-go-back sequence.

helps clarify that there is no infinite chain of processes being created, and that the child processes themselves create processes. A fourth approach is to ask students to classify or categorize ("which of the following if-statements will always execute?"). I recommend keeping the Beatty et al. (Beatty et al. 2006) article close at hand.

- Consider using paired, isomorphic questions (created a priori or on-the-fly) for critical course concepts or low correctness rates. (I can remember reviewing the basics of recursion before intending to move on to backtracking, and being shocked by the low correctness on a "review" ConcepTest. Instinctively, I began creating ConcepTest after ConcepTest until my conscience would allow me to proceed.) Much as you would like to move on, realize that the vast majority of students will be lost if you do so without remediation. Moreover, with PI, you will **know** when they are lost. It is common to cover slightly less material in a PI class as compared to a traditional lecture class (Crouch et al. 2007), but I think slightly less coverage is a small price to pay for increased conceptual mastery.
- Resist the temptation to short-circuit a PI iteration when you see a high correctness rate on the solo vote. Even with 90% correctness, 10% of students could likely benefit from the group discussion. Others who answered correctly may have guessed. Of course, if students are consistently getting 90%-95% correctness on the solo vote, your questions are too easy. I try to keep my solo correctness around 50%. Use metrics to iteratively improve your PI implementation!
- Similar advice applies if you plan to use isomorphic questions. Even if correctness on the group vote is convincing, it is possible that students will not be able to stretch understanding across situational boundaries. As demonstrated in (Porter, Bailey-Lee, et al. 2011), isomorphic questions can give you confidence that students really do (or do not) exhibit understanding as concretized in the histograms.
- Display histograms only following the group vote, never between the individual vote and group vote. Otherwise, students may gauge the correct answer from the graph, influencing their subsequent discussion and vote. Keep this in mind especially when a solo vote has an alluringly high correctness rate, where viewing the graph would leave little doubt as to what to choose. Use **both** graphs when discussing the response options. For example, if distractor A had a higher response rate in the solo vote than group vote, use the solo graph to highlight the fact that many people initially chose A (so it is not "obviously wrong" as might be inferred from the group vote).

3.5 A Role for Existing Materials?

You may have taught a course several times, to the point where your lecture materials are high-quality artifacts where bugs have been incrementally washed away and explanations incrementally improved. What is the role for these materials in a PI lecture? My view is that they should most certainly be kept, in order to serve two important roles.

First, you might like your slides more than your textbook. Perhaps your slides are so comprehensive **because** you can't find a single, suitable textbook. Rather than or in addition to giving students reading quizzes on textbook sections, consider your "old" lecture slides as pre-lecture reading material.

Second, use your old slides as templates for your PI replacements. Logically divide an existing lecture into segments, each of which tests one or several closely-related concepts. Replace each such batch of slides with one or two mini-lecture slides and a ConcepTest. The mini-lecture slides should quickly summarize the more comprehensive material you are replacing. (You may feel better about "cutting" material like this if students have access to your old lectures.)

3.6 Incremental PI

It is likely that the features of PI — reading quizzes, ConcepTests, group discussion — should be adopted wholesale in order to capitalize on all of PI's affordances. But sometimes doing so can feel quite daunting. Is there time to update everything before the semester starts? Are students really going to understand the reading quiz material with no instructor guidance? How much material can I cover now that time is taken by ConcepTests?

Having taught OS once using standard lectures, I felt somewhat queasy as I was getting ready to teach a subsequent PI offering. There was so much to cover in the allotted 24 hours of lecture that I doubted whether I could cover it all using the PI model. And my lack of experience teaching OS meant that, while I knew which topics were important, I really didn't have much knowledge in the way of student difficulties and misconceptions. I was reasonably confident I could create effective ConcepTests, but I knew I couldn't afford to "waste" too many ConcepTests on aspects with which students had little trouble. At some level, creating ConcepTests is iterative: this one's too easy, this one was poorly-worded, etc. But I wasn't ready to take on all of this at once.

I decided that I would keep my lectures the same, but introduce reading quizzes. My intention was to learn about student difficulties through the reading quizzes, in order to shape the ConcepTests I would create for the next offering of the course.

Instead, what I found was that the reading quiz responses alone were cause for lecture restructuring and reorganization. I ended up shaping my lectures around difficulties uncovered by the reading quizzes, and found myself skipping slides whose points had already been discussed.

To prepare the next offering, I feel fortunate in two ways: first, that I have the reading quizzes ready; and second, that I have data (in the form of their reading quiz responses) with which to create ConcepTests grounded in actual student experiences.

The alternative two-step implementation path, of course, is to institute ConcepTests but not reading quizzes. Reflecting on this alternative, I worry that without reading quizzes, mini-lectures will not suffice to get students ready for ConcepTests. There likely isn't time to add ConcepTests in the middle of otherwise standard lectures.

4 Conclusion

Research evidence for PI in CS, as well as its adoption rate, is building. ConcepTests are becoming available for more and more core courses. I hope the above research synthesis, implementation tips and anecdotes provided here give you sufficient reason and confidence to give PI a try. I encourage you to share your PI experiences with the larger CS-education community, corroborating or tempering what is currently believed. I am excited by the growing possibility that we have found a research-proven, educationally-beneficial method for teaching CS.

References

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.

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

Crouch, C. H., Watkins, J., Fagen, A. P., & Mazur, E. (2007). Peer instruction: Engaging students one-on-one, all at once. In E. F. Redish & P. J. Cooney (Eds.), *Research-based reform of university physics.* American Association of Physics Teachers.

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 August 19, 2011).

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

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

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

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.

Porter, L., Bailey Lee, C., Simon, B., Cutts, Q., & Zingaro, D. (2011). Experience report: a multi-classroom report on the value of peer instruction. In *Proceedings of the 16th annual joint conference on innovation and technology in computer science education* (pp. 138–142). New York, NY, USA: ACM.

Porter, L., Bailey-Lee, C., Simon, B., & Zingaro, D. (2011). Peer instruction: Do students really learn from peer discussion in computing? In *ICER '11: Proceedings of the seventh international workshop on computing education research.* New York, NY, USA: ACM.

Reay, N. W., Li, P., & Bao, L. (2008). Testing a new voting machine question methodology. *American Journal of Physics*, *76*, 171-178.

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, 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, D. (2010a). Experience report: Peer instruction in remedial computer science. In *Ed-Media 2010: Proceedings of the 22nd world conference on educational multimedia, hypermedia & telecommunications* (pp. 5030–5035). AACE.

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