Think-Pair-Share in a Large CS1 Class: Does Learning Really Happen?

Aditi Kothiyal
Inter-Disciplinary Program in Educational Technology
IIT Bombay, India
aditi.kothiyal@iitb.ac.in

Sahana Murthy
Inter-Disciplinary Program in Educational Technology
IIT Bombay, India
sahanamurthy@iitb.ac.in

Sridhar Iyer
Department of Computer Science and Engineering
IIT Bombay, India
sri@iitb.ac.in

ABSTRACT
Think-pair-share (TPS) is a classroom active learning strategy in which students work on activities, first individually, then in pairs and finally as the whole class. TPS allows students to express their reasoning, reflect on their understanding and obtain prompt feedback on their learning. While TPS is recommended to foster classroom engagement and learning, there is a lack of research based evidence in computer science education on the benefits of TPS for learning. In this study, we investigate the learning effectiveness of TPS in a CS1 course. We performed a quasi-experimental study and found that students who learned via TPS performed significantly better on a post-test than students who learned the same concept via lecture. We also conducted a survey and focus group interviews to understand student perceptions of learning with TPS. The majority of students agreed that TPS activities helped improve their conceptual understanding. From an instructor's point of view, TPS was useful to address the challenges of a large class, such as students tuning out or getting distracted and was easy to implement even in a large class.

Categories and Subject Descriptors
K.3.2 [Computers and Education]

Keywords
CS1, large class, active learning, think-pair-share, experimental study, effectiveness, learning.

1. INTRODUCTION
CS1 is an introductory programming course at many universities. Typical goals of a CS1 course are conceptual understanding of programming constructs, code tracing to predict the output, debugging and modifying code, and finally writing the program itself [7]. At our institute, CS1 is mandatory for freshman engineering students of all disciplines. It is a large class (450 students) with diversity in terms of student prior exposure to programming and motivation. Hence an instructor faces the challenge of ensuring that students across these variations are engaged and learning effectively.

Active learning techniques are known to enhance student engagement and improve student learning [13] . The choice of the active learning technique to be used for a class or a topic depends upon the corresponding instructional goals. For CS1 we need an active learning technique that is easily implementable in a large classroom setting and can be used for the goals stated above. Think-Pair-Share (TPS) [9], [10] is one such active learning technique. TPS is a structured co-operative strategy implemented in three phases as follows: 1) Think: The instructor poses a question and students think and write their answer to it, 2) Pair: Students work in pairs on an extension of the task posed in the Think phase and 3) Share: Students share their solutions and engage in a class-wide discussion, moderated by the instructor.

TPS has several known benefits of small group cooperative learning, such as engaging students with the content, the instructor and each other [1] [3], development of higher order thinking skills [4], and opportunity for formative assessment [2]. Despite the benefits of TPS, it has not been researched and evaluated in computer science education (CSE) for evidence of student learning. In a related paper [5], we provided evidence that TPS results in high student engagement in a large CS1 class. In this paper, we present a two group experiment showing that TPS results in effective learning. In addition we also offer guidelines on how to design TPS activities that not only meet the CS1 instructional goals but also are easy to implement in a large class.

Our broad research goal was to study the effects of TPS activities on the conceptual understanding and application of CS1 concepts in a large enrolment class. Our specific research questions (RQs) were:

1) Do TPS activities lead to increased conceptual understanding and application of CS1 concepts?
2) What are the students' perceptions of learning with TPS?
3) What are the instructor's perceptions of teaching with TPS?

We performed a control group experiment to answer RQ1, surveyed students and conducted focus group interviews to answer RQ2 and used instructor class logs to answer RQ3. We found that students in the experimental group who learned a concept via TPS performed significantly better on the post-test than students in the control group who learned the same concept via an interactive lecture. Further, in the perception survey, a majority of the students agreed that they would not have learned as much from the lectures had there been no TPS activities, and that the tasks in each of the phases helped conceptual understanding. Instructor perception confirmed that many benefits of TPS activities, such as getting students to engage deeply with the content, continue to hold even in a large CS1 classroom setting.

2. BACKGROUND AND RELATED WORK
Active learning comprises research-based instructional strategies in which students engage with the content in a deeper way than

ITICSE '14, June 21–25, 2014, Uppsala, Sweden.

http://dx.doi.org/10.1145/2591708.2591739 .

just listening to lecture or copying notes. They express their thinking and reasoning by writing, speaking, drawing diagrams, and problem-solving [1][13]. Characteristics of active learning methods are that students often work in small groups, are engaged in tasks emphasizing qualitative reasoning and conceptual thinking, and receive rapid feedback [13].

Active learning methods researched in CSE include pair programming (PP) [12], [14] peer instruction [15], [16], just-in-time teaching [11], process oriented guided inquiry learning [6] and inverted classroom [8]. PP [12], [14] is a collaborative technique in which two students work together in the lab to solve open programming problems like design, development and testing [18]. It has been shown that PP improves student retention and confidence [12] and quality of programs produced [18]. A method for large classes that been extensively researched in CSE is Peer Instruction (PI) [15]. In PI, students work on multiple choice questions aimed at improving conceptual understanding and qualitative reasoning [15]. It has been shown that students learning via PI have higher grades in the CS0 course than equivalent students learning via a traditional lecture [16].

An active learning strategy that has not received significant attention in the CSER community is Think-Pair-Share (TPS). TPS and PI have some common features. In both methods students initially think about the problem posed by the instructor individually and commit to an answer, the difference being that students record a written answer in TPS and vote on their choice in PI. Students then discuss in pairs or groups. In TPS, this discussion can involve a checking of each others' answers, as well as working together to solve the next part of the problem; while in PI, the discussion is mostly focused on the students' votes. The final Share phase in TPS can be compared to the Whole-class discussion recommended in PI [19]. In this phase, TPS involves discussion of multiple solutions and their pros and cons, while PI focuses on students' reasoning for various answers. Both methods can be used to address the goals of conceptual understanding. In addition, TPS allows the posing of open-ended problems such as writing programs, which is not possible with PI.

TPS is based upon several key ideas that have been shown to be effective for learning, including active learning [13] and cooperative learning [4]. TPS has been shown to be a good classroom formative assessment technique [2][17]. Since grouping is done informally, the constraint on movement and requirement of teaching assistants, typical of large classes, is overcome. While there have been some studies to establish the effectiveness of TPS for learning in domains like psychology [2][17], there are fewer studies evaluating the effectiveness of TPS in the learning of computer science and other STEM disciplines. As we showed in a related paper, >80% of students were engaged and on-task in each phase of the TPS [5].

3. COURSE DESIGN

3.1 Course Goals and Challenges

The specific instructional goals of our CS1 class were to teach our freshman engineering class programming concepts and skills, i.e., conceptual understanding of programming constructs, analysis of a program to predict the output and debug/modify code, developing programming logic to solve a specific problem, writing pseudo code and finally writing the program itself.

All classes were taught by the same instructor. Challenges that the instructor had to deal with included: large number of students (450); large diversity in prior exposure, ranging from students who had never used a computer to those who were fluent in C++ programming; and stadium style seating with fixed chairs and tables, leading to constraint on student movement and grouping for collaborative learning. The challenge was to keep students engaged with the content, the instructor, and with each other, despite these constraints.

3.2 General Course Implementation Details

The CS1 course was conducted over 14 weeks in Spring 2013. The topics covered were control structures such as conditionals, iteration, functions and recursion, data structures such as arrays, matrices, strings and queues, object-oriented structures such as classes and the concept of inheritance. In the first two weeks of the course, Scratch was used to introduce basic programming constructs; the rest of the course was taught via C++.

Students were from varied engineering disciplines but not CS majors. They were divided into two sections for lectures. Each section had two 90-minute interactive lectures per week, in which instructor lecturing was interspersed liberally with instructor and student questions, open discussions, student activities and program demos. The course did not have recitations and problem solving activities were included into the lecture itself. The course also consisted of a 2-hour lab per week, which consisted of programming exercises designed to give students practice in the application of the skills and concepts learned in the lectures.

3.3 Instructional Method: Think-Pair-Share

Problems addressing goals of tracing, modifying and writing code can have multiple valid solutions. As instructors, we want students to not only be able to devise a solution to the problem, but also analyze the pros and cons of various solutions. Hence we need a format of active learning that: (i) gets students vested in the problem by getting them to first devise their own idea of the solution, (ii) prevents students from feeling daunted with the task by allowing them to work with each other, and (iii) affords discussion of pros and cons of multiple solutions. The three phases in TPS offer a natural fit to meet these requirements. Most lectures had two TPS activities on average.

Think phase. The instructor presented the task, and students worked individually on the task for about two minutes and wrote their answers in their notebooks. For example, the instructor presented the following question (Also see Table 1).

"Predict the output of the following program:

```
int main() {
    int A[4], *p;
    for (int i = 0; i < 4; i++) A[i] = i;
    p = &A[0];
    cout << *p << " " << *(p +=2) << *(p+1) + *(p-1) <<
endl; }"
```

Pair phase. The instructor gave a task related to or extended from the Think phase question. In the above example the task was, "Check your neighbor's solution and determine if it is the same as yours. If not, discuss and come up with a solution that you both agree on." The students worked with their neighbors to complete the task in three to five minutes. The instructor walked along the aisles, encouraging discussion and answering queries.

Share phase. The instructor led a class-wide discussion related to the tasks in the Think and Pair phases. In the above example, the instructor elicited a few responses, and then executed the program to show the output. He then asked students to propose

modifications in the code that could result in the other responses that came up. Students followed the discussion to verify their solution and discuss 'what-if' scenarios. This phase was open-ended, lasting from three to ten minutes depending on the depth of the discussion. At an appropriate point, the instructor transitioned from this phase into the next topic.

During the first TPS activity of the semester, the instructor described the structure of the activity to the students and what was expected of them. Thereafter, we found that the problem statement was sufficient to cue the students to the task. For the first few activities, the instructor explicitly encouraged the students to write their responses during the Think phase. In subsequent activities the students did not need any prompts for any of the phases.

3.4 Creation of TPS activities

Once it was decided to use TPS in the course, the instructor piloted a few TPS activities in the class and observed students' behavior along with an external observer. Their observations led to the following design principles for TPS activities which were employed throughout the rest of the semester:

1. Each phase of think-pair-share should be meaningful in solving the problem. That is, the problem must contain parts that require individual thinking and writing, the Pair phase deliverable should require two students to work together, and the Share phase activity should merit a class-wide discussion.
2. The Think and Pair phases should have precise deliverables to ensure that the ensuing Share phase discussion is focused towards the answer for the original problem.
3. The phases should be logically connected. Students should use the output of one phase in next phase.
4. Sufficient time should be planned for each phase. Too little time can cause frustration among students and too much time can lead to boredom.

TPS can be used for a variety of learning outcomes. Depending on how the phases of the activity are designed, TPS can be used to improve students' ability to analyze a given program, write programs for the given tasks or acquire conceptual knowledge of programming constructs. A summary of the structure of the TPS activities for various learning outcomes is shown in Table 1.

Table 1: Examples TPS activities

Instructional goals	Think Pair Share	Example as shown in the slide to students
Conceptual understanding	*Think* Students write down the answer to the given question *Pair* Students (i) Identify parts of the answer that they have missed out. (ii) Discuss their answers; do pros-cons analysis if there are multiple solutions. *Share* Instructor discusses (i) What are all the essential parts in the answer? (ii) Pros-cons of various solutions given by students	"Consider an unsorted array of N elements. Think: Write the pseudo code for sorting the array Pair: Discuss your answer with your neighbor, do pros and cons analysis of your algorithms Share: Follow instructor led discussion of your solutions and others." *This led to a discussion of various sorting algorithms.

Code tracing: Predict the output; Debug/modify the given code	*Think* Students determine and write down the answer.
	Pair Students (i) check each others' solution (ii) discuss changes in code needed to get others' solutions
	Share Instructor (i) executes the program and shows the output (ii) discusses a few modifications based on student answers.
		"Predict the output of the following program:
		`int a = 1, b = 2, c = 3;`
		`int* p, int* q;`
		`p = &a; q = &b;`
		`c = *p; p = q;`
		`*p = 13;`
		`cout << a << b << c << endl;`
		`cout << *p << *q << endl;`"
		Think: Draw the memory arrangement and predict output.
		Pair*:* Check your neighbor's solution. If you don't agree, discuss and come up with a solution that you both agree upon.
		Share*:* See demo of above code and modified versions."
		*The example for the outcome "Debug/modify" is similar

Develop programming logic for a problem: Write program.	*Think* Students write down the pseudo-code. *Pair* Students (i) identify missing pieces in each others' solutions (ii) write the program. *Share* Instructor (i) shows one possible solution. (ii) Discusses a few representative student solutions.	"Recall your program to reverse a 4 digit number. Extend your solution to arbitrary integers. Think: Write the pseudo-code individually. Pair: Write the C++ code with a partner. Share: Compare your solution with demo10-reverseNum-mod1.cpp"
Design a solution: Write pseudo-code	*Think* Students write down the different parts (structures and functions) of the solution *Pair* Students discuss the pseudo-code for other structures and functions that are required *Share* Instructor discusses a few representative solutions.	"Design a taxi scheduling service for an airport as follows: (i) When a driver arrives, his ID is entered in an array (ii) When a customer arrives the earliest waiting driver is assigned Think: What structures and variables are required? Pair: Discuss the pseudo-code for the functions that are required. Share: Follow instructor led discussion of your solutions and others."

4. RESEARCH METHODOLOGY

4.1 Learning outcome measurement

To answer RQ1, "Do TPS activities lead to increased conceptual understanding and application of CS1 concepts?" we conducted a two group pre-post quasi-experimental study to determine the effectiveness of TPS activities over interactive lecture.

Sample. One of the two sections was randomly assigned as the experimental group (263 students), which received a TPS treatment, and the other as control group (184 students), which received a regular interactive lecture. The equivalence of both the groups was established on the basis of a pre-test which had 5 questions testing students' understanding of prerequisite concepts. The results of a Mann-Whitney U test between the pre-test scores of the experimental group (M_{expt}*=16.3, SD=5.6),* and the control group *(*$M_{control}$*=16.7, SD=6.7)* showed no significant difference (Mann Whitney U = 20440, *p=0.574).*

Procedure. The concept chosen for the experiment was the interleaving of multiple threads in the CPU. This concept is new to both novices and advanced learners, so their prior knowledge does not play a role. The instructor chose to use Scratch to explain threads because it is a visual programming environment in which multi-threading is very easy to implement. In both the groups the instructor first explained the concept of multiple threads and thread synchronization via an interactive lecture. Next the instructor presented a problem on interleaving of threads.

"Consider three threads as shown below.

Thread A	**Thread B**	**Thread C**
When Run flag clicked, Say "Thread A start"; Repeat 2 times ● Move 10 steps; Say "Thread A done"	When Run flag clicked Say "Thread B start"; Turn 90 degrees; Broadcast "event"; Say "Thread B done";	When I receive "event", Glide to (0,0).

Assume that: (i) 'When' and 'Say' statements result in 2 assembly instructions, (ii) Loop initialization, increment and condition check, each results in 1 assembly instruction, and (iii) all other statements result in 3 assembly instructions. Also assume that: (a) all assembly instructions are atomic and take the same amount of time, (b) CPU time-slice is sufficient for 3 assembly instructions. What are the possible interleaved execution sequences?"

In the control group, the instructor explained the solution as a worked example while students followed along and asked questions. In the experimental group, the problem was presented as the following TPS activity:

> "Think: Write one possible interleaved execution sequence.
> Pair: Check your neighbours' solution. If it is the same as yours, come up with a second interleaved execution sequence.
> Share: Instructor explains one possible solution and discusses alternate solutions."

Post-test. The post-test consisted of a single question (this was sufficient because it covered the entire concept that was taught using the TPS activity) on thread interleaving similar to the one above. It was included as the last part of the quiz that students took in the class following the above problem-solving activity. The post-test question was graded out of a maximum score of 4.

4.2 Student perception survey and focus group

To answer RQ2 "What are students' perceptions of learning with TPS?" we administered a survey to all students. The instrument had questions on student engagement and learning. All questions were on a 5-point Likert scale (strongly disagree, disagree, neutral, agree, and strongly agree). The questions relevant to the learning construct were:

> Q1. Thinking about the problem and writing the solution during the think phase helped me learn CS1 concepts.
>
> Q2. Discussing my solution with my partner during the pair phase helped me learn CS1 concepts.
>
> Q3. Listening to other students' solutions and discussion during the share phase helped me learn CS1 concepts.
>
> Q4. I would not have learned as much from the lecture if there had been no think-pair-share activities.

In addition, at the end of the course, we conducted four focus group interviews with 8-10 students in each group. The interviews lasted 30 minutes each and were conducted by an external observer. The interviews were audio recorded, transcribed and analyzed using the content analysis technique.

4.2 Instructor perception data

To answer RQ3, "What are the instructors' perceptions of teaching with TPS?" we have two sources of data. The instructor maintained detailed logs of the class. In addition, an external observer, who attended all classes, maintained notes of classroom observations.

5. RESULTS

5.1 TPS activity leads to increased conceptual understanding

250 students in the experimental group and 169 students in the control group took the post test. The distribution of scores was not Normal, hence we used Mann-Whitney U-test to compare means of the two groups, the results of which are shown in Table 2.

Table 2: Analysis of post test scores of experiment

Experimental Mean (SD)	Control Mean (SD)	p-value	Difference significant at p=0.05
1.91 (1.65)	0.88 (1.38)	0.00	Yes

We find that there was a statistically significant difference between the post-test scores of the two groups, with the experimental group (TPS) performing significantly better than the control group (interactive lecture). Further, Cohen's effect size (d = .67) suggests a moderate to high practical significance.

This experimental study was conducted in one class during which the applicability of TPS for the learning outcome of conceptual understanding was tested. In the interest of fairness to students we did not repeat the study in any further classes. For the remainder of the semester, both the sections were taught using interactive lectures interspersed with TPS activities for maximum learning in both sections. In Table 3, we present the scores of a problem from a course exam which was based on concepts taught to both groups using TPS. We find that there is no significant difference between the group means when both learnt via the same method. This result continues to hold for all exam problems throughout the semester. Table 3 also shows the final exam scores of students, where we find no significant difference between the groups. These results together indicate that it was the introduction of the TPS activity which caused the significant difference between the post test scores of the two groups.

Table 3: Comparing groups when taught via the same method

	Experimental Group Mean (SD)	Control Group Mean (SD)	p-value	Difference significant at 0.05?
Problem taught via TPS (out of 4)	3.66 (1.69)	3.43 (2.07)	0.848	No
Final exam (out of 100)	64.08 (23.7)	66.48 (23.44)	0.18	No

To understand these results better we classified the students based on their pre-test scores into 3 categories, low (<40%), medium (40-70%) and high (>70%) achievers. We similarly classified them based on their post-test scores to low (0), medium (1 or 2) and high (3 or 4) achievers. We determined the percentage of students who transitioned from category A pre-test to category B post-test by counting the number of students who were in category A pre-test and category B post-test. This enabled us to develop an empirical model of the students' learning in each group as shown in Figures 1 & 2.

Our first observation is that while nearly two-thirds of the control group got zero on the post-test problem a majority of the students in the experimental group scored three marks or higher. Next we observe that in the experimental group 61% of high achievers remain high achievers, while a significant percentage of medium (37%) and low achievers (30%) move into a higher achievement category. In the control group, however, 79% of high achievers moved into low or medium achievement categories and small percentages of medium (14%) and low (18%) achievers moved into higher achievement categories. This demonstrates that the TPS activity enabled students of all categories in the experimental group to perform better on the post-test as compared to the students in the control group.

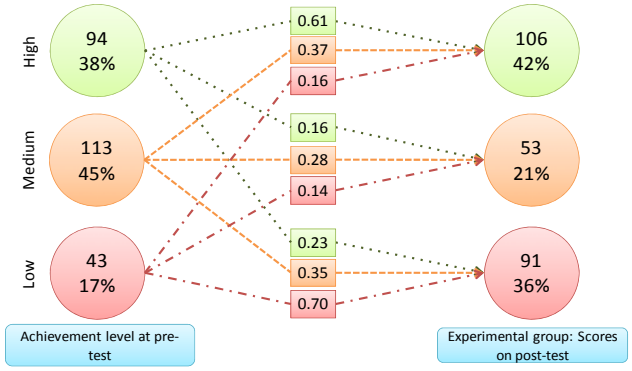

Figure 1: Transition diagram of experimental group

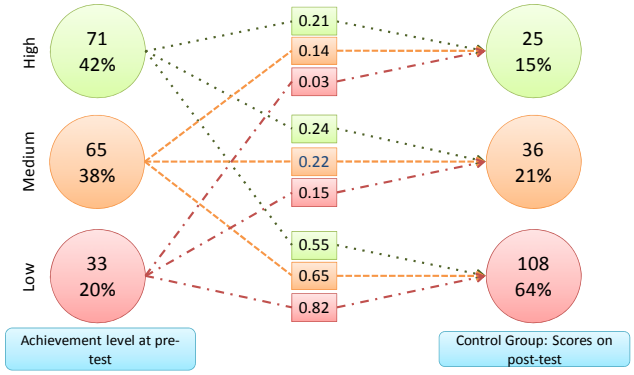

Figure 2: Transition diagram of control group

5.2 Students perceive TPS useful for learning

We received 336 valid responses to the student perception survey. The summary of student responses is presented in the Table 4.

Table 4: Student perception of learning with TPS

	Strongly Agree + Agree (%)	Neutral (%)	Strongly Disagree + Disagree (%)
Q1	72	21	7
Q2	67	24	9
Q3	73	21	6
Q4	58	29	13

These results show that a majority of the students perceived each stage of the TPS activities to be useful for learning CS1 concepts. A majority of the students also felt that they learned better from an interactive lecture interspersed with TPS activities than an interactive lecture only. Transcripts of the focus group interviews were coded, categorized and classified to identify student perceptions regarding learning with TPS and confirm the survey results. In the interest of space, we are not reporting all the results of the content analysis and only a few illustrative quotes below.

"The think and pair parts were equally important. Unless we think on our own, we won't get to know at what level we are. When we were made to think on certain questions we realize that these are some places we get stuck. We discuss those things with our partner, we realize that he overcame this problem in a certain manner and then we may come up with better solutions.."

"In a class of 240 you can come with 4 or 5 different solutions. […] In that half an hour [of TPS] we are able to learn five methods of solving a problem and pros and cons of each method. That's more that you can learn in an hour."

Finally, the overall percentage of the end of semester course evaluation conducted by our institution was 85%, which is comparable to the top courses at our institution.

5.3 Instructor finds TPS engaging for all students

The instructor's perceptions of the benefits and challenges of teaching with TPS were as follows:

1) TPS is useful to address the challenges of students tuning out, getting distracted or going off-task. When specific deliverables were given in each stage of the activities, students were on-task. For this to happen, the activities must be interesting and balance student ability and challenge. Such activities ensured that the problem of boredom and frustration was resolved.
2) The activities were easy to implement even under the constraint of fixed seats. Students naturally turned to their classmates on their left and right, formed informal groups and discussed their solutions.
3) The activities easily scale to large numbers. The Think and Pair phases are distributed among the students and so do not pose a challenge to the instructor. The Share phase can be a bottleneck, but the instructor did not find it so because many solutions turned out to be similar and so only the first instance of each type of solution needed to be discussed.
4) TPS mitigates problems due to diversity of prior knowledge. Since seating and pairing are random, learners without prior knowledge often get benefits of one-on-one tutoring. Learners with prior knowledge of a given topic are engaged due to discussion with peers or tutoring.

5) There is increased participation by everyone, not just the vocal students. Since everyone has worked on the problems individually and in small groups, everyone has something to contribute to the share phase and so gets involved.
6) The entire class gets the benefits of multiple and unusual solutions because the instructor explicitly invited sharing of those solutions which were different from what had already been discussed.

6. DISCUSSION AND CONCLUSION

Our first research question "Do TPS activities lead to increased conceptual understanding and application of CS1 concepts?" was answered by the results of the quasi-experimental study which showed that the group who learned a concept via a TPS-activity performed significantly better (with a moderate to high effect size) than the group which learned the same concept from an interactive lecture. Further, the transition diagrams show that a majority of students in the experimental group transitioned into equal or higher performance level from pre to post test. In the control group however, students moved into lower performance levels.

One concern of such an experimental study is student motivation. Even though we established group equivalence on the basis of a pre-test, this was a mandatory course for all freshmen. Hence it is possible that the students who were new to programming were more enthusiastic about the course than others and so learned better. However since we chose the concept of threads using Scratch as the target concept for the study, it was a new concept to all students, and we expect that all students had the same motivation to learn this concept. Another concern is instructor bias. While it is possible that the subtle changes of instructor behavior between the two methods can impact student learning, the instructor made every effort to ensure that the interactive lecture was engaging. In addition the end-of-semester evaluations (84% vs. 86%) show that the two sections did not perceive differences in instructor behavior.

Results of the second research question "What are students' perception of learning from TPS?" showed that a majority of students approved of a TPS-based classroom environment, and they would not have been able to learn CS1 as well had they not performed the TPS activities. The results of our final research question "What are the instructor's perceptions of teaching with TPS?" have shown that the instructor perceives TPS i) to be an effective technique that engages all students of varying levels, and ii) is easy to implement even in a large class.

Think-Pair-Share has been known to be an effective strategy for improving learning outcomes in various disciplines [2][17]. Our study has reconfirmed this finding in a CS1 large class. The main takeaway for instructors is that rather than framing a question as an open discussion to the whole class, creating a TPS activity is more effective. One reason is that the TPS activity ensures that students are vested in the outcome in each phase leading up to the discussion. The structured phases focus the discussion and ensure that it is more fruitful than an open discussion which typically tends to be dominated by the vocal students. The three phase structure also ensures that there is some part of the activity to keep different students engaged, thus addressing the issue of diversity of achievement levels. The guidelines and examples we provide in Table 1 help an instructor operationalize TPS for a programming course. This paper thus provides another effective active learning technique for CS instructors of large classes.

7. REFERENCES

[1] Bonwell, C. C., and Eison, J. A. Active learning: Creating excitement in the classroom. Washington, DC: School of Education and Human Development, George Washington University, 1991.

[2] Butler, A., Phillmann, K. and Smart, L. Active learning within a lecture: Assessing the impact of short, in-class writing exercises. *Teaching of Psychology*, 28 (4), 257-259.

[3] Cooper, J. L. and Robinson, P. Getting Started: Informal Small-Group Strategies in Large Classes. *New Directions for Teaching and Learning,* 81.

[4] Kagan, S. The structural approach to cooperative learning. *Educational Leadership,* 47(4), 12-15.

[5] Kothiyal, A., Majumdar, R., Murthy, S. and Iyer, S. "Effect of Think-Pair-Share in a large CS1 class: 83% sustained engagement", In *Proc. 9th Int. Comp. Edu. Research Workshop*, August 12-14, 2013, San Diego, USA.

[6] Kussmaul, C. Process oriented guided inquiry learning (POGIL) for computer science. in *Proc. 43rd ACM Tech. Symp. on Comp. Sci. Edu.*, pp. 373-378.

[7] Lee, C. B. Experience report: CS1 in MATLAB for non-majors, with media computation and peer instruction. In *Proc. 44th ACM Tech. Symp. on Comp. Sci. Edu,* pp. 35-40.

[8] Lockwood, K. and Esselstein, R. The inverted classroom and the CS curriculum. In *Proc. 44th ACM Tech. Symp. on Comp. Sci. Edu,* pp. 113-118.

[9] Lyman, F. The responsive classroom discussion. in Anderson, A. S. ed. *Mainstreaming Digest,* College Park, MD: University of Maryland College of Education, 1981.

[10] Lyman, F. Think-Pair-Share: An expanding teaching technique, *MAA-CIE Cooperative News,* v. 1, pp 1-2.

[11] Martinez, A. Using JITT in a database course. In *Proc. 43rd ACM Tech. Symp. on Comp. Sci. Edu,* pp. 367-372.

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

[13] Meltzer, D. E. and Thornton, R. Resource Letter ALIP–1: Active-Learning Instruction in Physics, *Am. J. Phys.,* 80, 6.

[14] Nosek, J. T. The case for collaborative programming. *Comm. of the ACM,* 41.3 pp. 105-108.

[15] Porter, L., Lee, C. B., Simon, B., and Zingaro, D. Peer instruction: do students really learn from peer discussion in computing? In *Proc. 7th Int. Workshop on Computing Edu. Research*, pp. 45-52.

[16] Simon, B., Parris, J., and Spacco, J. How we teach impacts student learning: Peer instruction vs. lecture in CS0. In Proc. *44th ACM Tech. Symp. on Comp. Sci. Edu*, pp. 41-46.

[17] Vreven, D. and McFadden S. An Empirical Assessment of Cooperative Groups in Large, Time-compressed, Introductory Courses. *Innov. High Edu,* 32, 85–92.

[18] Williams, L., Kessler, R. R., Cunningham, W. and Jeffries, R. Strengthening the case for pair programming. *Software, IEEE,* 17(4), 19-25

[19] http://www.cwsei.ubc.ca/resources/files/Clicker_guide_CWSEI_CU-SEI.pdf