Problem Posing Exercises (PPE): An instructional strategy for learning of complex material in introductory programming courses

Shitanshu Mishra
Educational Technology - IDP
Indian Institute of Technology Bombay
Mumbai, India
shitanshu@iitb.ac.in

Sridhar Iyer
Department of Computer Science and Engineering
Indian Institute of Technology Bombay
Mumbai, India
sri@iitb.ac.in

Abstract -

Problem posing refers to the generation of a new problem or a question, by a learner of a given topic. It has been shown to be an effective strategy for learning of complex material in domains such as mathematics. In this study we have defined Problem Posing Exercises (PPE) as a problem posing based instructional strategy to scaffold the learning of complex material and problem solving in a first year undergraduate computer science course (CS1). We implemented PPE in a large CS1 course of 450 students and found that PPE has positive effects on students' learning and engagement. We also examined the differences in effect of PPE for novice and advanced learners and found that PPE has more evident effects on novices than advanced learners.

Keywords - Problem posing; problem posing exercises; introductory programming course; complex material learning; problem solving; instructional strategy.

I. INTRODUCTION

Problem posing refers to the generation of a new problem or a question based on the given situation [4]. For example, a student may generate a question based on the content of a lecture, homework, tutorial, or lab. Problem Posing (PP) has been shown to be an effective instructional strategy for teaching–learning of complex materials and training students on problem solving [1], in many domains such as mathematics [2] and reading comprehension [3].

Learning of complex materials and problem solving are important goals for introductory programming course (CS1). However, there is very little work on the use and effectiveness of PP in the context of teaching programming courses. In this study we have defined Problem Posing Exercises (PPE) as a PP based instructional strategy to scaffold the learning of complex material and problem solving in CS1. During PPE, we instructed students to generate two problems (and their corresponding solutions) for topics that they learnt in the previous few classes.

We investigated the effects of PPE on students' learning achievements and engagement. Our research questions (RQs) are:

RQ1. How does PPE affect learning of first year engineering students in CS1 course?
This RQ has been further refined to sub-RQs as:
   a. How does PPE affect students' perception of learning in CS1 course?
   b. How does PPE affect students learning outcome (as measured by tests) in CS1 course?

RQ2. What is the effect of PPE on student engagement in a CS1 course?
   a. What is the effect of PPE on student engagement with the topics in a CS1 course?
   b. What is the effect of PPE on student engagement with the activity itself?

RQ3. What is the transition pattern from - pretest performance - to - posttest performance of novice and advanced learners, as a result of the PPE activity.

To answer the above research questions we conducted a field study of incorporating PPE as an instructional strategy in a large CS1 class. Our intervention was introduced in lab sessions as a regular lab activity, for five days, to five different batches. Each batch performed the PPE activity once, for about one hour. Students belonged to different departments, and none of them was from Computer Science. Students were free to generate problems of any form, such as "write a program", "debug", "predict the output", or any other subjective form.

Students' learning (RQ1) was assessed using their performances in in-semester quiz (pretest) and mid-term exam (posttest), which were scheduled just before and after the PPE intervention week. Test scores were stratified to give three levels (low, medium, and high) of performances. We administered a survey questionnaire to determine students' perceptions of the usefulness of PPE for their learning (RQ1) and engagement (RQ2).

Quality of generated problems, operationalized as the difficulty levels of those problems, were also evaluated as low, medium and high levels. This data was analyzed to determine how the quality of problems generated by students during PPE is related to pretest and posttest performances (RQ3).

Our results show that PPE has positive effects on students' learning: high probability (82%) for students to transit to same or higher level of performance (pretest to posttest). Student perception data also triangulated this result, as number of students who felt that PPE would improve students' learning was high. We found a similar trend for PPE's effect on students' engagement. We also

978-0-7695-5141-8/13 $31.00 © 2013 IEEE
DOI 10.1109/T4E.2013.45

found that the performance of advanced learners was independent of PPE intervention, and PPE appeared to be effective for Novices only.

Section II discusses theoretical foundation of our paper, and gives an account of other research works from literature which are related to our research. Section III gives a detail of course logistics and implementation of PPE intervention. Section IV elaborates our research methodology. Section V presents our results obtained. Section VI is the discussion, and section VII presents conclusions of our study and future scope of this research.

II. THEORY AND RELATED WORK

In this section, we first discuss the theoretical basis for using Problem Posing as an instructional strategy (II-A) from the perspective of learning complex material and problem solving in computer programming. We then focus on prior work on the use of PP as an instructional strategy (II-B). We conclude by mentioning a few alternatives to PP that have been used to scaffold the learning of complex materials and problem solving (II-C). A detailed account of the mechanisms and features of PP can be found in [5] and [1].

A. Theoretical Basis for using Problem Posing

Students in an introductory programming course need to simultaneously learn complex programming concepts [6], and synthesize them together to actually solve any programming problem. These parallel demands could lead to cognitive overload [7] and disrupt the learning process. For the smooth execution of learning process, instructional strategies which can scaffold learning of complex materials, along with training students on problem solving, are needed.

A number of studies in cognitive science give evidence that problem posing is a fundamental component in cognitive processes that operate at deep conceptual levels, such as the learning of complex material [1]. Problem posing is an instructional strategy, in which students are naturally or explicitly made to engage in dialog with the instructor or with peers, and present their queries or analytics in the form of questions. Problem posing has been applied through many models as an instructional strategy [8].

B. Application of Problem Posing strategy

Silver [2, 9] used problem posing for teaching mathematics, and considered problem posing as one of the way of creativity enriched mathematics education. Crespo [10] and Toluk-Uçar [4] investigated the effect of problem posing on pre-service primary teachers' in the domain of mathematics, and found that problem posing is an effective instructional strategy for teaching mathematics to pre-service teachers. Barlow and Cates [11] recommended that problem posing be incorporated as an instructional strategy into all professional and undergraduate education programs.

Wong [3] investigated the effectiveness of problem posing as an instructional strategy, designed to improve students' processing of prose and found that problem posing had successful effects on student prose processing. King [12] compares problem posing strategy, as 'self-questioning', with two other strategies of 'summarizing', and 'note taking-review'. Self-questioners performed better, on a retention test, than the summarizers and note-taking-reviewers. Graesser and Person [1] documented the problems posed during tutorial sessions in research methods and algebra courses, and has discussed the relevance of problem posing to theories of learning, cognition, and conversation.

C. Need for our work

There are few strategies to scaffold learning of complex materials [13] and problem solving [14, 15, 16] in a variety of domains. Some strategies specific to Computer Science education are given in [17], [18], [19], and [20]. Ala-Mutka [20] mentions use of visualization to teach complex concepts of computer programming. Resnick et al. [18] and Meerbaum-Salant et al. [19] have used Scratch as a visual programming language to teach complex programming concepts. Liu et al. [17] have used simulation games to teach problem solving in computer programming. However there is a dearth of literature on incorporating problem posing as an instructional strategy in Computer Science education, and more specifically in teaching programming at CS1 level.

III. COURSE FORMAT AND IMPLEMENTATION

The setting for our study was a large enrollment CS1 class of 450 first year undergraduate students, across various engineering disciplines, excluding CS majors; the student characteristics are described in Section IV-A. The goal of the CS1 course was to teach introductory programming using Scratch and C++ as programming language platforms. The course was conducted over 14 weeks in Spring 2013. The components of course implementation included lectures, labs, project, PPE, and assessments. PPE implementation was done in the 5th week-lab.

Topics covered till the 4th week of the course were: Sequences, variables, operators, arrays, jump constructs, loop constructs, threads (in scratch), and events (in scratch). There was a 30 marks quiz in the end of 4th week, and another 50 marks examination in the start of 6th week. We have used these two tests as pretest and posttest respectively.

A. What did students do?

PPE was a strange activity for students and this strangeness could have obstructed smooth cognitive responses from students. Therefore to bring students to a comfort zone, we implemented a “collaborative” PPE in which two students collaborate as a team to generate problems. Each pair was asked to generate two questions, pertaining to the topics covered so far. They were free to set either a programming problem or a conceptual question, and had to submit detailed answers to their generated questions. Students were given motivation that 18 best questions from each lab-batch would be selected as the practice questions for the next lab-batch, and questions could be selected for the mid-semester question paper. Students were given only one open-ended guideline “The questions should be challenging but should not be too difficult for the students in the next batch to complete in the lab”. The time given to generate two problems was 45 Minutes, but for many students time was

extended up to an hour. Students submitted their generated problems over Moodle [21], the learning management system used in the course.

B. What did Teaching Assistants(TAs) do?

A team of TAs was assigned to talk to students and motivate them to brainstorm and generate problems that may lead to deeper application of the concepts taught in the class. There was one TA per 10 students and 90 students per lab session. TAs were told to intervene whenever they found any student stuck in the activity, or sitting idle for long time or, busy doing some out-of-context work. It was the responsibility of a senior TA to coordinate with junior TAs to manage all the logistics in the lab session.

C. What did the researcher and the instructor do?

Researcher laid guidelines for students and TAs during PPE. He further administered a student perception survey after the posttest. At the end of each session, the researcher took open ended feedback from two students, chosen randomly. Instructor administered the Quiz (pretest) and mid-semester Exam (posttest) as a part of regular semester requirements.

D. Samples of questions generated

Following are some instances of problems generated by students during PPE:

Low difficulty questions:

i. "Write a program to get the next 10 numbers on entering a number."
ii. "Write a C++ program to find out whether a given number is odd or even."

Medium difficulty questions:

iii. "Write a program in cpp to obtain following output up to N digits. User will input N.

1
23
456…"

iv. "Write the program of e=1+1/1!+1/2!+1/3!+.....for given n terms."

High difficulty questions:

v. "Write a program that calculates the sum of the first n entries of a series whose difference is in arithmetic progression. The first 3 entries of the series are given."
vi. "Write a program to construct a Pascal's triangle for n number of rows."

The difficulty levels, shown in the above examples, were evaluated using the evaluation rubrics, mentioned in the section IV-B-2. For example: If we compare [i] and [v], we find that [v] requires large number of logical considerations, and its solution requires knowledge of arithmetic Sequence, in addition to the understanding of programming concepts, such as loops, whereas [i] poses less logical challenge, and have straight forward solution.

IV. STUDY METHODOLOGY

Our research goal is to investigate the effect of PPE as an instructional strategy in a CS1 course. To elaborate more, our first research question (RQ1) relates to the effect of PPE on the learning of programming concepts by first year engineering undergraduate learners. Our second research question (RQ2) relates to the effect of PPE on students engagement, and our third research question (RQ3) relates to determining how the quality of questions generated during PPE correspond with test performance.

We have used a combination of quantitative and qualitative tools, and analyses instruments. They include pretest-posttest performances, qualitative analysis of questions generated by students during PPE, quantitative analyses of survey data, and a strata transition diagram.

A. Sample

The PPE intervention was given during lab sessions, which were scheduled every weekday (from Monday to Friday). We were able to collect the experiment data on only three out of five days due to logistics issues. Therefore out of total 450 students registered for the course, we got opportunity to conduct experiment on 270 students under our observation. All were first year students majoring in different branches of engineering (Chemical Engineering, Mechanical Engineering, Electrical Engineering, etc.). Students who were admitted to the institute were among the highest ranked in an extremely competitive exam testing analytical skills in mathematics, physics and chemistry (they were all among the top 1000 out of 500000 students). Hence all students in the study can be considered as equivalent in all respects, except prior exposure to programming.

To address RQ3, we needed to determine prior programming background; hence we used the demographic survey conducted by the course instructor in the first week of the semester. Students had to respond to a series of questions on their familiarity with computers and programming experience. Some of these questions were:

1. I have taken Java/C++ in X standard (Yes / No),
2. I have taken Java/C++ in XII standard (Yes / No),
3. I have written programs using advanced constructs beyond what is taught in X / XII standard (Yes / No),
4. I have participated in programming contests (Yes/ No), if so which one (text box).

We classified students who responded 'Yes' to any of these questions as 'advanced learners'. Those who had not taken a programming as a subject in any standard in school, or had not programmed outside of school were classified as 'novices'. 332 students completed the above survey, hence only those students were considered in the sample to address RQ3. Based on the survey results, we found that the number of novice learners was 217 and the number of advanced learners was 115.

As mentioned in section III-A, we conducted a "collaborative" PPE, in which two problems were generated by a pair of students. In this paper we consider one pair of students as one 'sample'. From 270 students, we had 135 samples (pairs). We considered the average of test scores of each pair to be the test score of that sample. Each pair

generated two problems together so we considered the aggregate quality of the two problems as the quality of problem posed by that sample. We identified a pair to be 'advanced' if at least one student of the pair was an advanced learner, else we identified the pair to be 'novice'.

B. Instruments.

We used a variety of quantitative and qualitative approaches and data collection instruments to triangulate our data. These are as follows:

1) Pretest and posttest scores.

The pretest was administered in the week before the PPE intervention and the posttest was administered in the week after the PPE. These scores were used to determine acquisition of basic programming concepts (RQ1b), and the correspondence between scores and quality of problems generated during PPE (RQ3). Both pretest and posttest contained questions based on Scratch and C++. There was a mix of conceptual and programming question types, including: 'predict the output', 'debug the program', as well as 'write a program'. These were typical CS1 questions. For example, a program for Bubble Sort with erroneous array initialization and loop conditions was given and students were required to debug the program. In another question, they were required to write a program to output the Fibonacci series.

Analysis

We compared students' performance in pretest and posttest. We also stratified the scores into various levels of achievements (Low: pretest < 40%, Medium: 40% <= pretest score <= 70%, High: pretest > 70%), and analyzed the pattern of transitions of student performances from pretest to posttest.

2) Problems Generated during PPE..

Each student pair was asked to generate two questions. By the end of the whole exercise there were 270 questions, from 135 samples.

Analysis

We analyzed each generated problem on 3 different quality parameters: 1. Difficulty level of the question, 2. Computational thinking concepts [6], targeted by the question, and 3. Creativity, which is based on how much has the questioner succeeded to bring a new context in the programming question.

For the difficulty analysis, we developed evaluation rubrics to rate the difficulty level of the question as high, medium, or low. These three levels were defined as follows:

- Low: Problems with well understood logic and straightforward solution.
- Medium: Problems with some amount of logical challenge and does not have straightforward solution.
- High: Problems which are highly logically challenging and have no straight forward solution.

In the rubrics, levels are distinguished using two parameters, "logic" and "straight forwardness of solution". Examples given in section III-D clarify the meaning of these two parameters. These definitions were frozen after consulting 3 experts, who are research scholars and skilled in both computer science and education research methodology; each of them had minimum 17 years of programming experience. Each of them analyzed a given set of 20 problems before they were consulted for their definitions of these rubric items.

Then we analyzed each question generated on the basis of the programming concepts targeted by it. These concepts were chosen from computational thinking concepts described in [6] and include: Sequence, Loops, Threads, Events, Conditionals, Operators, Variables, and Arrays. Finally, we analyzed each question on the basis of their creativity level. In this paper we have used only difficulty level as the metric to operationalize the quality of questions generated.

3) Student perception questionnaire.

We created a survey to address RQ1a: How does PPE affect students' perception of learning in CS1 course?, and RQ2: What is the effect of PPE on student engagement with the topics in a CS1 course and with the activity itself? All of the questions in the survey were on a Strongly Agree to Strongly Disagree Likert scale. To elicit honest feedback from students the complete survey was anonymous. There were 4 questions, having target statements as follows:

Q1: "The question generation activity helped me in mid-semester exam."

Q2: "Question generation activity helped me in improving my deep understanding of topics that I studied in the class".

Q3: "Question Generation activity helped me making CS101 interesting."

Q4: "I would like to repeat this activity by generating more questions for the next quiz."

Analysis

Quantitative analysis of all four questions gives the measure of students' perception of learning and their perception about usefulness of PPE for their engagement.

V. RESULTS

A. Perception of learning.

Fig. 1 shows the perception of students about the statement, "The Question Generation activity helped me in preparing for posttest." It came out that 42% of students agree or strongly agree to the fact that PPE helped students to get prepared for the posttest. Only 22% of 332 disagreed or strongly disagreed. So, ignoring neutral responses (36%), it is evident that majority perceive PPE to be helpful in learning. This answers RQ1a: What is the effect of PPE on perception of learning in CS1 course of first year engineering students?

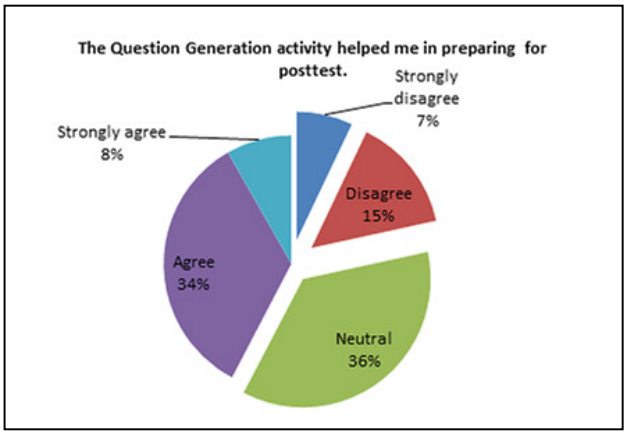

Figure 1. Survey Question 1. Total no. of participants=332

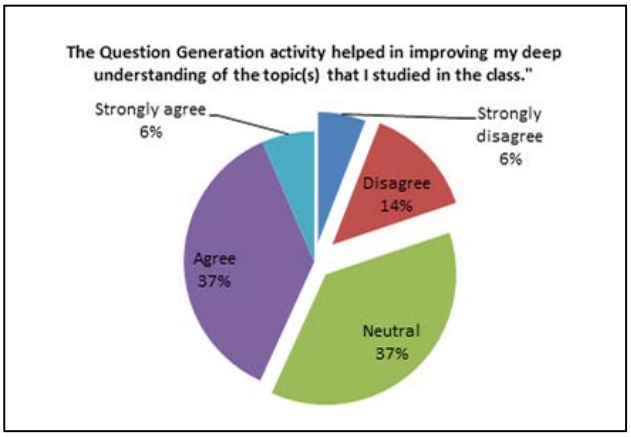

Figure 2. Survey Question 2. Total no. of participants=332

To triangulate the above result, fig. 2 shows the pie chart for the reply to "The Question Generation activity helped in improving my deep understanding of the topic(s) that I studied in the class." In this case also, 43% students agree that PPE helped in improving their learning, and only 20% disagreed. More specifically, fig. 2 results indicate that PPE helps in deeper learning.

B. Students' learning outcome

Table [I] shows difference of scores between posttest and pretest for different groups of students generating low level, medium level, and high level of questions. Out of 135 samples, 32 have generated low level difficulty questions, 83 have generated medium level difficulty question, and 20 have generated high level difficulty question.

The table [I] shows that there are slight posttest gain in all the three levels and the highest posttest gain is achieved by those students who have generated Medium level of difficult problems. We note that none of the three differences of means are significant.

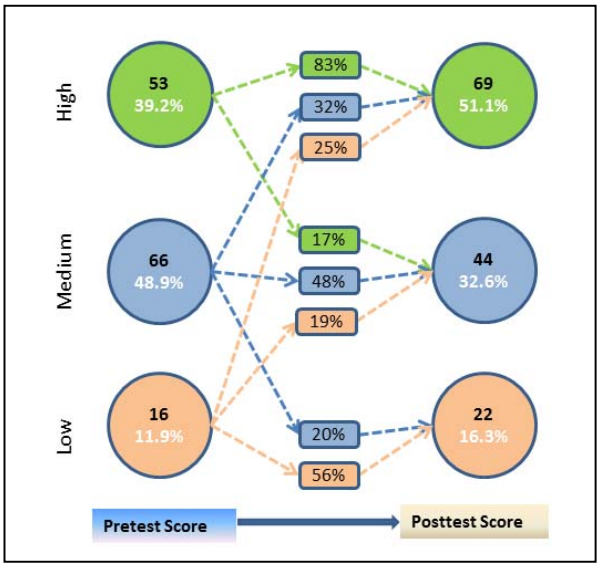

Figure 3. Two – Layer Strata Transition Diagram. Total samples=135

TABLE I. DIFFERENCE OF SCORES [POSTTEST – PRETEST] FOR DIFFERENT GROUPS OF STUDENTS, GENERATING LOW LEVEL, MEDIUM LEVEL, AND HIGH LEVEL OF QUESTIONS

Quality (Difficulty) of Generated-problem	***Level 1 (Low), N=32***	***Level 2 (Medium), N=83***	***Level 3 (High), N=20***
Mean of [Post-Pre] Scores (out of 100)	1.04	7.59	4.83

Although we did not get any significant recommendations in the form of posttest gains, yet the result encouraged us to analyze into more depth. We stratified posttest and pretest scores into three levels (Low, Medium, and High), as described in the section IV-B-1, and then obtained a transition diagram from different strata of pretest to different strata of posttest. Fig. 3 shows the Strata Transition Diagram. In the diagram, low, medium, or, high levels correspond to levels of scores in pretest and posttest. Numbers inside circles show the number of samples who have achieved low (peach color), medium (blue color), or, high (green color) scores in pretest or posttest. Transition probabilities from any i^{th} strata of pretest score to any j^{th} strata of posttest score are written inside the rectangular boxes, on the edges joining the i^{th} pretest strata to the j^{th} posttest strata.

As seen from the strata transition diagram, probability that any student remains in the same performance level or improves his performance is significantly higher than the probability that he/she reduces his/her performance. In 112 samples out of total 135 samples, it is evident that students improved or retained their performance. Even if we subtract the number of samples where students have retained their low performance level, there are 103 out of 135 samples that students have retained or increased their performance levels. This shows that there is a trend of performance gain from pretest to posttest learning outcome (RQ1b).

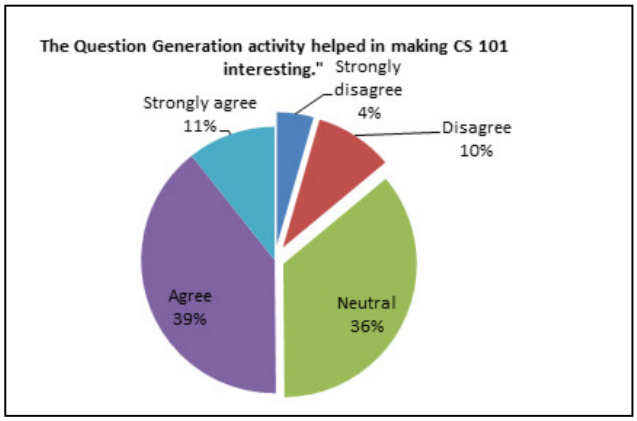

Figure 4. Survey Question 3. Total no. of participants=332

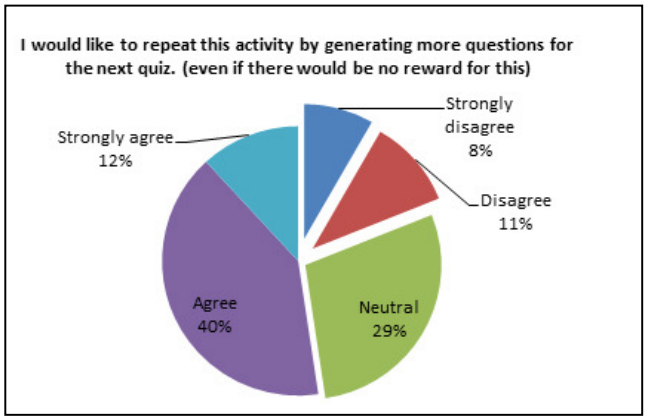

Figure 5. Survey Question 4. Total no. of participants=332

C. *Students' Engagement*

Fig. 4 shows the perception of students about the statement, "The Question Generation activity helped in making CS 1 interesting" (RQ2a). It came out that 50% out of 332 students agree or strongly agree to the fact that PPE helped students to make CS1 interesting. Only 14% students disagreed or strongly disagreed. So, it is evident that majority perceive PPE to be helpful in improving the students' engagement.

The Q4 of the survey (fig. 5) indicates that PPE not just makes CS1 interesting but itself is of interest to students (RQ2b). 52% of total 332 students agree or strongly agree to the statement that they would like to repeat PPE by generating more questions for the next quiz, even if there would be no reward for the activity. Only 19% of the students disagree or strongly disagree.

D. *Transition pattern from - pretest performance - to - posttest performance – via - different quality levels.*

To study the transition pattern of samples from different strata of pretest scores - to- different strata of posttest performance – via – different quality levels of generated problems (RQ3), we have used a 3-layer strata transition diagram as shown in Fig 6. The first layer corresponds to three different strata of scores in pretest, the second layer corresponds to the three difficulty levels of generated problem, and the third layer is corresponding to the three strata of posttest scores. Pretest and posttest scores were stratified with the same rule as mentioned in section IV-B-1. Interpretation of circles and color coding are the same as mentioned in section V-B.

We have analyzed these patterns separately for advanced learners and novice learners. Out of total 135 samples, 62 were found to be advanced learners, and 60 were novice learners.

1) *Transition pattern for Advanced Learners*

Fig. 6 shows the transition probabilities from pretest score strata to difficulty level categories and from difficulty level categories to posttest score strata. The 3-layered strata transition diagram has been analyzed in two passes. In the first pass (PASS1) we derive inferences with the help of transition probabilities between pretest score strata layer and difficulty levels layer, and in the second pass (PASS2) we make inferences based on the transition probabilities between difficulty levels layer and posttest score strata layer. Following interesting transition patterns are evident for advanced learners, viz.

i. Probability of generating medium level difficulty questions by both high and medium level pretest performers is fairly high: 55% and 69% respectively. Whereas low level pretest performer struggled to generate medium level problems (PASS1).

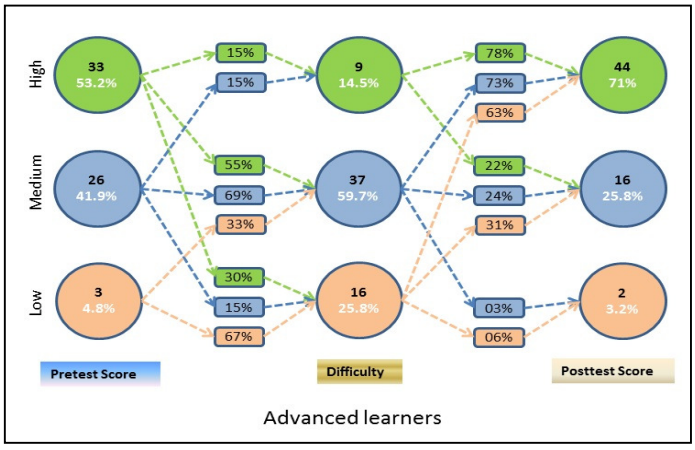

Figure 6. Three – Layer Strata Transition Diagram. Total Advanced Learner Samples=62

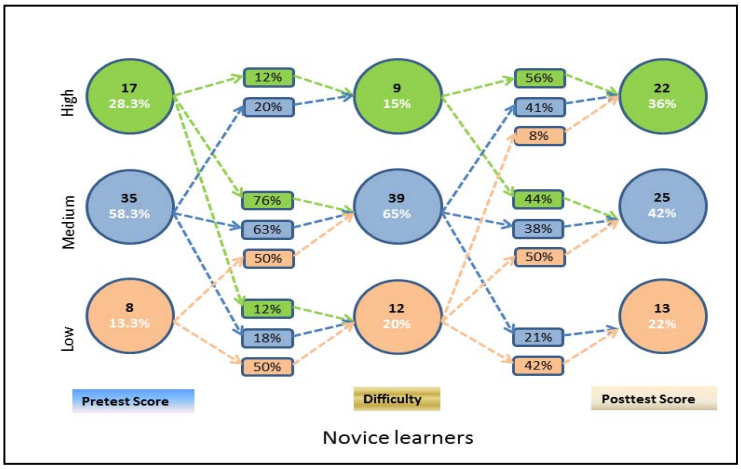

Figure 7. Three – Layer Strata Transition Diagram. Total Novice Learner Samples=60

ii. For all levels of difficulty – high, medium and low, probabilities of transitioning to 'High' level of posttest performance is very high, which is 78%, 73%, 63% respectively. This suggests that advanced learners have high posttest performance irrespective of their performance (difficulty level) in the PPE (PASS2).

2) Transition pattern for Novices

Fig. 7 shows the transition probabilities from pretest score strata to difficulty level categories and from difficulty level categories to posttest score strata. This 3-layered strata transition diagram has also been analyzed in the same manner as the previous transition patterns in the case of advanced learners. Following are the interesting transition patterns for novice learners:

i. As compared to any other difficulty level, probabilities of generating medium level difficulty problems by all 'high', 'medium', and 'low' level pretest performers is more, which are 76%, 63%, and, 50% respectively (PASS1).

ii. Transition to 'high' level of posttest performance is more evident behavior of 'high' difficulty problem generators. Whereas Students who generate 'medium' difficulty problems have almost equal probability to perform 'high', or 'medium in the posttest (PASS2).

3) Novice and advanced learners: comparison

i. Probability of generating medium level difficulty questions by both high and medium level pretest performers is evident in both novice and advance cases (PASS1).

ii. High probability of generating low difficulty questions by high pretest performers is evident in the case of advance learners only (PASS1).

iii. For both novice and advance learners: Higher the difficulty level of generated problem, higher is the posttest score, but it does NOT necessarily imply that: lower difficulty yields lower posttest performance (PASS2).

iv. Advanced learners show high tendency to raise their performance, more than novice learners (PASS2).

v. Interestingly, both Novice and advance learners have shown zero and no probability to transit to low posttest performance if they have generated high difficulty levels of questions (PASS2).

VI. DISCUSSION

We first investigated effectiveness of PPE in students' learning of basic programming concepts (RQ1). Student perception data show that 42% of students agreed to the positive effect of PPE on their learning, while 36% were neutral and only 22% disagreed. On the other hand, from the analysis of posttest and pretest scores we are not able draw any significant inference, as the posttest-pretest difference appeared to be insignificant. This may be due to the fact that we have used the course quizzes, setup by the instructor, as our pre and post assessment tools. Although they were based on the same course content, the posttest was of higher difficulty level than the pretest, as this was the instructor's intention. If the pretest and posttest were equivalent, we may have found significant differences, thereby leading to an inference that generating problems of moderate difficulty level results in high learning gains. On the same note, transition probabilities from pretest to posttest, establish that there was a high trend of transition from lower to higher performance levels, than higher to lower performance. This triangulates the positive effect of PPE on students' learning.

When we analyze students' engagement (RQ2) in terms of how does PPE influence students' interest, we find that the majority of students felt that PPE made CS1 interesting (50% agree, 36% neutral). Moreover, student perception data also support that PPE itself was interesting for students and majority of them were willing to repeat the exercise in future (52% agree, 29% neutral).

Further, we investigated the mechanism that how well does any low, medium or high performer (pretest) generates problem during PPE, and how does the quality of his/her

problem generation affects his/her posttest performance (RQ3). We infer that majority of students, irrespective of being advanced or novice learner, generated problems of medium level difficulty. One interesting pattern about advanced learners was that, they are highly probable to generate low level difficulty problem than novice learners.

In addition to this, advanced learners show that irrespective of the quality of problem they generate, most of them tend to score high. On the other hand for novice learners, quality of problems generated seems to have correspondence with their performance in posttest. Therefore, we recommend that PPE is more effective for novices than advanced learners.

VII. CONCLUSION

We conducted a field study of incorporating PPE as an instructional strategy in a large CS1 class. The objective of the study was to investigate how does the PPE intervention affects students' learning and engagement and in what pattern does its affects varies for advanced learners and the novices? We conclude that the PPE as an instructional strategy in an introductory programming course has: (i) Positive effects on students' learning of basic programming concepts. (ii) Positive effect on students' engagement with the course. (iii) More usefulness in the case of novice learners than advanced learners.

We also conclude that PPE as an instructional strategy is an effective technique and needs to be further investigated for teaching-learning of computer programming. Some research questions which now arise are: (i) How can we make PPE more effective for teaching introductory programming course? (ii) In what form can PPE be made a better strategy for advanced learners? and (iii) What is the effect of PPE in other computer science courses? All these constitute the foundation of our future work.

REFERENCES

[1] A. C. Graesser and N. K. Person, "Question asking during tutoring", American educational research journal, vol.31.1, 1994, pp. 104-137.

[2] E. A. Silver, "Fostering creativity through instruction rich in mathematical problem solving and problem posing", ZDM, vol. 29.3, 1997, pp.75-80.

[3] B. Y. Wong, "Self-questioning instructional research: A review", Review of Educational Research, vol. 55.2, 1985, pp. 227-268.

[4] Z. Toluk-Uçar, "Developing pre-service teachers understanding of fractions through problem posing", Teaching and Teacher Education, vol. 25.1, 2009, pp. 166-175.

[5] S. I. Brown, and M. I. Walter, The art of problem posing, Routledge, 2004.

[6] K. Brennan, and M. Resnick, "New frameworks for studying and assessing the development of computational thinking", In Proceedings of the 2012 annual meeting of the American Educational Research Association, Vancouver, Canada , 2012.

[7] F. Paas, 2004, A. Renkl, and J. Sweller, "Cognitive load theory and instructional design: Recent developments", Educational psychologist, vol. 38.1, 2003, pp. 1-4.

[8] N. Wallerstein, "Problem-posing education: Freire's method for transformation", Freire for the classroom: A sourcebook for liberatory teaching, 1987, pp. 33-44.

[9] E. A. Silver, "On mathem pp.atical problem posing", For the learning of mathematics, vol. 14.1, 1994, pp. 19-28.

[10] S. Crespo, "Learning to pose mathematical problems: Exploring changes in preservice teachers' practices", Educational Studies in Mathematics, 52.3, 2003, pp. 243-270.

[11] A. T. Barlow, and J. M. Cates, "The impact of problem posing on elementary teachers' beliefs about mathematics and mathematics teaching", School Science and Mathematics, vol. 106(2), 2006, pp. 64-73.

[12] A. King, "Comparison of self-questioning, summarizing, and notetaking-review as strategies for learning from lectures", American Educational Research Journal, 29(2), 1992, pp. 303-323.

[13] J. J. Van Merrienboer, and J. Sweller, "Cognitive load theory and complex learning: Recent developments and future directions", Educational psychology review, vol. 17.2, 2005, pp. 147-177.

[14] P. C. Wankat, and F. S., Teaching engineering, New York: McGraw-Hill Oreovicz, 1993, pp. 269-280.

[15] D. H. Jonassen, "Instructional design models for well-structured and III-structured problem-solving learning outcomes", Educational Technology Research and Development, vol. 45(1), 1997, pp. 65-94.

[16] M. C. Kim, and M. J. Hannafin, "Scaffolding problem solving in technology-enhanced learning environments (TELEs): Bridging research and theory with practice", Computers and Education, vol. 56.2, 2011, pp. 403-417.

[17] C. C. Liu, Y. B. Cheng, and C. W. Huang, "The effect of simulation games on the learning of computational problem solving", Computers and Education, vol. 57.3, 2011, pp. 1907-1918.

[18] M. Resnick, J. Maloney, A. Monroy-Hernández, N. Rusk, E. Eastmond, K. Brennan, and Y. Kafai, "Scratch: programming for all", Communications of the ACM, vol. 52.11, 2009, pp. 60-67.

[19] O. Meerbaum-Salant, M. Armoni, and M. M. Ben-Ari,August, "Learning computer science concepts with scratch", In Proceedings of the Sixth international workshop on Computing education research,m ACM, 2010, pp. 69-76.

[20] K. Ala-Mutka, "Problems in learning and teaching programming." Codewitz Needs Analysis, 2012.

[21] M. Dougiamas, and P. Taylor, "Moodle: Using learning communities to create an open source course management system", In World conference on educational multimedia, hypermedia and telecommunications, Vol. 1, 2003, pp. 171-178.