Effectiveness of Using Guided Peer Code Review to Support Learning of Programming Concepts in CS2 Course: A Pilot Study

Dr. Tamaike Brown, State University of New York at Oswego

Assistant Professor of Computer Science, Department of Computer Science, State University of New York at Oswego

Dr. Gursimran Singh Walia, Georgia Southern University

Gursimran S. Walia is Professor of Computer Science at Georgia Southern University. His main research interests include empirical software engineering, software engineering education, human factors in software engineering, and software quality. He is a member of the IEEE Computer Society. Contact him at gwalia@georgiasouthern.edu

Mr. Alex David Radermacher

Alex Radermacher is a lecturer at North Dakota State University. He teaches introductory programming courses as well as upper-level software engineering courses including the department's Senior Capstone Design course. His research interests include student learning, software testing, and software development processes.

Dr. MANINDER SINGH, ST CLOUD STATE UNIVERSITY
Dr. Mourya Reddy Narasareddy gari, Rider University

Effectiveness of Using Guided Peer Code Review to Support Learning of Programming Concepts in CS2 Course: A Pilot Study

1. Introduction

This section provides a brief statement of the problem, research motivation, our proposed solution and high-level contributions.

Problem statement: Researchers at North Dakota State University (NDSU) have discovered that students enrolled in CS1/CS2 courses struggle at understanding programming concepts, make common mistakes that are recurring and don't think about the code quality as they learn to program. Researchers have discussed exploring *program comprehension* (helping students understand how a program works) and guiding their learning towards *program generation* (creating a part or whole program) [12]. In a programming domain, program comprehension tasks can include modifying a program's functionality, identifying incorrect behavior, or simply answering questions about the program. While program comprehension has been extensively studied in industrial settings, not much educational research can be found on the topic [1, 13]. Based on the empirical studies in industry, a common approach involves a programmer reading through the code line-by-line, which allows programmers to build their knowledge base methodically [1, 13]. A specific type of method called peer code review (PCR) has been found to be useful at not just building high-level abstractions but also enabling knowledge sharing among programmers. PCR is a well-defined quality assurance activity that is designed for the sharing of knowledge and for the purpose of improving code quality [1, 2, 3]. Classified at the evaluation level of Bloom's taxonomy [4], PCR involves students systematically checking (mainly by viewing and reading) parts of someone else's source code for errors, with the aim of creating better code when they develop their own.

Research motivation: Our initial study supports the idea that PCR can be used to support programming pedagogy [14] in higher education. Based on a feasibility study, it was found that finding errors in someone else's code reduced the likelihood of committing similar errors when students went on to develop their own code. However, students were not reporting many errors when they did not have a list of questions that guides them to identify errors, which resulted in students not pay attention to detail when reading someone else's code. Students indicated that they needed more guidance on what types of errors to look for when performing the code review. Including some set of expectations that can help students map those expectations to the source code under review.

Research objective: We aimed to empirically investigate whether guided PCR could be effective in teaching CS2 students programming concepts. Additionally, we investigated the possibility of helping students to retain information (from PCR sessions) to times when they develop their own code. Furthermore, we also wanted to investigate if pair-based review can help with knowledge

retention or sharing among students. To accomplish this, guided PCR sessions were conducted where students were asked to follow a provided checklist in order to systematically check for programming errors when reading source code on line-to-line basis.

Research methodology: We designed and implemented four different PCR sessions over the course of four weeks in one CS2 classroom. During each week, students were given a piece of code covering a specific data structure and were asked to review and find errors in the code. The provided code pieces were seeded with five categories of errors: initialization/declaration, method call/ definition, array/linkedList/trees/, output and flow of control. We analyzed data gathered from the guided PCR sessions, reflection sessions conducted after each PCR session, work conducted by students (assignments, quizzes, and exams completed as part of the course), and a feedback survey conducted at the end of study.

Contribution: This paper provides empirical evidence of the impact the guided PCR had on CS2 students' understanding of programming concepts. While prior research has examined the effects of PCR on students' motivation to learn and course attendance, it lacks evidence concerning the usefulness of guided PCR at improving students' conceptual knowledge in their courses. We also provide evidence on how PCR can be used to enable knowledge sharing among students and foster in-class discussion. We also hope that this study will lead to the development of a more robust error checklist and a larger repository of programs that are seeded with realistic defects and can be used by other CS1/CS2 instructors.

Outline: The rest of the paper is organized as follows: Section 2, looks at related work. Section 3 presents the experimental design implementation details used by the authors of this paper. Following this, the data collection and analysis of the results are presented in Section 4. Section 5 contains a discussion of the results. Finally, Section 6 concludes the paper.

2. Related Work

This section focus on peer code review (PCR) and its application to computer science and software engineering education. Code review has its origin from software engineering and has been a common programming practice for anyone who has worked in software development professionally. It was first introduced in 1975 followed by multiple studies on its feasibility and effectiveness in industry [6, 7]. This has led to educators showing interest in incorporating PCR in their introductory CS course. Educators have been using PCR as an educational tool in different ways and a number of desired learning outcomes have been noted [10, 11].

In a PCR experiment designed and conducted by Trytten et al., the instructor selected one student assignment for peers to review and critique. The process involved students answering a questionnaire consisted of 15 questions that evaluated their understanding of their peers' code. At the end of the questionnaire, students shared responses with each other and discussed the reason for the choice they selected. The result of the study shows that the students lab attendance was improved [8].

In another peer code review experiment, Li et al., implemented a code review process in a 3rd year undergraduate web application development course. In this study, students were expected to use their coding standard knowledge and learn peer review skills from the experiment. Students selected their own reviews and the review process was done face to face. The aim of the experiment was to assess coding quality when students used code review in programming assignments [9]. Survey was used to collect data from students. The survey results of the study showed that students were motivated to learn coding standards when using code review and it facilitated communication among the students [9].

In a prior study conducted at North Dakota State University, 19 CS1 students participated in a PCR study. The aim of the study was to evaluate the usefulness of PCR in teaching introductory programming concepts in CS1 course. The experiment involved students viewing a reading code developed by someone else to find and record errors using a reflection sheet. The result of the study showed that students who participated in a PCR session were able to understand hard to comprehend CS1 programming concepts and make fewer programming errors when developing their own code [14]. Through the end of the study some topics were still of concerns to students however, the study showed improvement in students learning of hard to comprehend topics over the course of the experiment [14].

While the above research documented positive results on code review there is no evidence pointing to the use of a checklist that will guide students during the review process. Our work differs from prior study in that it attempts to empirically evaluate guided PCR w.r.t to students' knowledge acquisition of introductory programming concepts in CS2 course. This work improves on prior study conducted at North Dakota State University by including a checklist in PCR.

3. Experimental Design

This section provides details regarding the major goals, lists research questions, study procedures that were followed to accomplish the goal, and the data that was collected during this study. Two runs of the experiment were conducted. We refer to the initial run as a pilot study that helped us modify the second run of the experiment design.

A. Study Goal

The essential goal of this research was to evaluate the usefulness of guided (checklist) PCR on CS2 students' learning with respect to the programming concepts of data structures and algorithms. More specifically, we wanted to understand if checklist-based PCR could guide students in understanding and overcoming common programming mistakes identified in the CS2 course. To accomplish these goals, four sessions of guided PCR study were conducted in one CS2 class wherein code samples were externally developed and seeded with faults (in consultation with the CS2 instructor). Students were asked to review code samples and log faults using the PCR checklist. Following each guided PCR session, students were asked to reflect on their review results, discuss errors, and ways to avoid them when developing their own

programs. At the end of study, students provided feedback on the usefulness of guided PCR in an introductory programming course.

B. Research Questions

The following research questions were formulated to accomplish the research goal:

- **Research Question 1 (RQ1):** *Which type of errors do students identify during guided PCR and why?*
- **Research Question 2 (RQ2):** *What is the effect of guided PCR on students learning of programming concepts in CS2 course?*
- **Research Question 3 (RQ3):** *What perceptions do students have on guided PCR and using it in CS2 course?*

C. Participating Subjects

This study was conducted in CS2 programming course at North Dakota State University. Sixteen students elected to participate in the study. The study sample was made up of CS and non-CS majors, most whom completed CS1 or some other previous introductory programming course.

D. Study Procedure

The study was conducted in one CS2 class over a period of four weeks during the summer of 2019. PCR sessions were conducted once per week and lasted approximately thirty-five minutes. During each PCR session, students were given a different piece of code to review. Each piece of code contained 20-70 source lines of code (SLOC) and was written in Java. Each code fragment was seeded with multiple errors representative of most commonly committed errors by students at North Dakota State University. No line of code exhibited multiple errors. The code examples were developed in consultation with two CS2 instructors at NDSU and included topics that students in CS2 had previously discussed in lecture including doubly linked lists, array sorting, trees traversal, Breadth First Search in graphs). The study procedure included following major steps:

Step 1: Introduction to PCR: Students were introduced to guided PCR and the purpose of the study.

Step 2: Individual Peer Code Review: Students were given individual error checklists, example code, and error sheets for them to record errors in the code. Students were first asked to review a piece of source code individually using the checklist. A sample of questions used to guide code review is shown in Figure 1. The researchers in consultation with the course instructor developed four pieces of code for error abstraction, with each piece of code becoming

1. Are all loops correctly formed, with the appropriate initialization, increment and termination expressions?
2. Do Boolean functions return the correct value?
3. Has correct syntax used for logical operators?
4. Are values assign to the correct variable?
5. Are there any typographical error such as = = instead of = or != instead of !
6. Are variables mathematical calculation done correctly such as adding instead of subtracting?

Figure 1: Sample Java Checklist

progressively more difficult, both in terms of programming concepts covered and the number of SLOC. In each piece of code, five categories of errors were seeded with on average of 20 errors in total; at least one error belonging to each category was seeded in each piece of code. The seeding of errors was done with input from instructors who have identified major errors committed by students in CS2 programming course at NDSU. Table 1 shows the category of errors. The output of this step was 16 "individual error sheets" (one per student) that described the errors (including Line # in source code and error description) found during each code review session.

Table 1: Category of Errors Seeded in Code

Category of Errors	Definition of Errors
Initialization Declaration Error	Not creating or declaring an identifier used in program
Method call / Definition error	Not naming or invoking a method correctly
Arrays, linkedList and trees error	Not indexing an array correctly, not creating a node correctly or defining pointers correctly, assigning a value to the wrong variable
Output error	Grammatical errors in displayed output or improper statement termination or incorrect output formatting
Flow of control error	Improper looping, incorrect value return for Boolean function or incorrect use of syntax for logical operators, adding numbers instead of subtracting

Step 3: Group Peer Code Review: Students were randomly assigned to pairs (groups of two) to find and record any additional errors on a new error sheet (group error sheet) using the checklist. Students were asked to first discuss their individual results (found during Step 2) and only record new errors (not found by either of the students belonging to the pair). The goal of this step was to identify the importance of team based review activity. The output of this step was 8 "group error sheets" (one per group) that described the new errors.

Step 4: Code Reflection sheet: After students finished reviewing each piece of code, they were asked to fill out a reflection sheet for that code fragment that described actual seeded errors. Each reflection sheet (a total of four for each PCR sessions) described actual seeded errors in the code, the location of each error, asked students to comment if they did (or did not) locate the error, and to provide comment if they did not agree it was an error.

Step 5: Discussion: After each reflection period, and prior to subsequent guided PCR session, researchers discussed the seeded errors in the code artifact and used the reflection to encourage discussion of common programming errors.

Step 6: Survey: At the end of the four-week study, a survey was administered to gather students' feedback on the usefulness of checklist in PCR exercise and its impact on their improved understanding of programming concepts. Students responded on set of questions using a 5-point Likert scale.

E. Data Collection

We collected both quantitative and qualitative data during the study run. The quantitative data included the number and category of errors found both by students individually and by students working in pairs during each of the guided PCR session. False positive errors were removed prior to the analysis by comparing the reported errors against the true errors that were seeded in each piece of code. We also collected data regarding the number and type of errors overlooked during guided PCR sessions, students' performance on assignments and quizzes pre-, post- and in-between PCR sessions. These quantitative data were collected to allow us to analyze whether guided PCR had any effect on CS2 students' learning of data programming concepts (RQ1). In addition, it allowed us to determine if there was any improvement in students' performance throughout the CS2 course (RQ1).

The qualitative data collected during the study included student reflection reports, and responses on the end-of-study survey (completed at the conclusion of all PCR sessions). The qualitative data will allow us to answer RQ2.

4. Results and Analysis

This section presents the analysis of the data collected from the study and is organized around three research questions listed on the experiment design section of this paper.

Research Question 1 (RQ1): *Which type of errors do students identify during guided PCR and why?*

This research question aimed at evaluating the use of guided PCR as a teaching technique in CS2 classroom. We examined the number of true errors reported by students when using guided PCR to review each piece of code. This was done in order to identify errors students were able to report, errors students were not able to report, and the effect the reflection / discussion had on

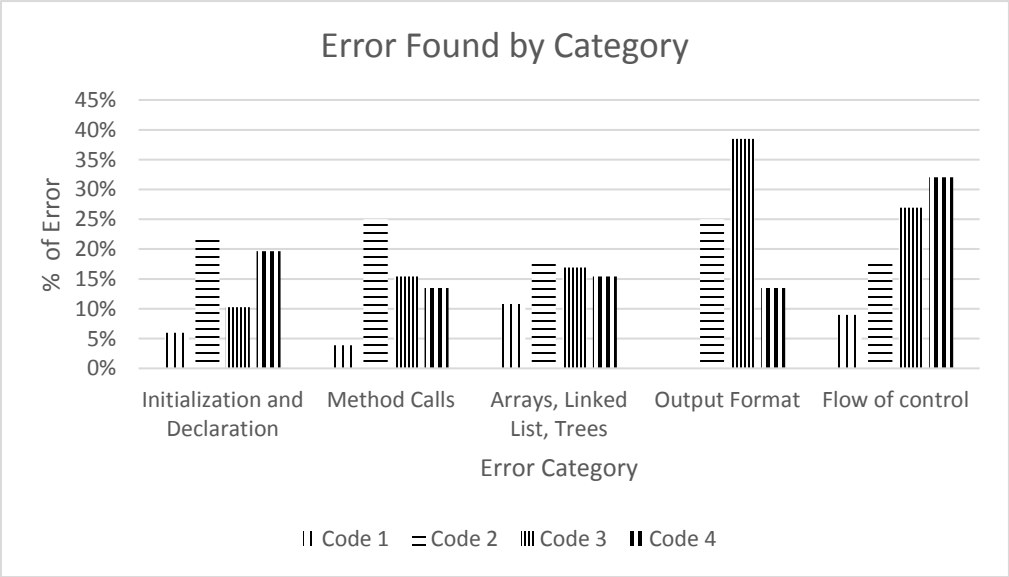

Figure 2: Percentage of Errors Reported

subsequent peer review performance. Figure 2 provides an overview of the results. To normalize the performance across each PCR session and error category, Figure 2 shows a comparison of the percentage (%) of error reported by students (when working individually) for each category of errors in each piece of code. Some of the major observations based on the analysis of data collected during the study run are discussed below:

Observation 1: The graph shows low error reporting percentage for Code 1 (doubly linked list). This was because the instruments for the pilot session of guided PCR did not accurately capture the error description or the location of the error reported by students. While these instruments include the error reporting sheet and the reflection sheet, students mentioned that they needed more guidance on how to use the checklist to find and record errors. The researchers used this feedback from students to improve the design of the previously mentioned instruments before conducting subsequent guided PCR sessions. In addition, based on students' comments from the error and reflection sheets, students found it hard to conform to the code review process and needed pointers on how to read checklist questions and compare them against the code under review. We are referring to the first PCR session as the pilot run. For subsequent PCR sessions, we included extensive training and developed more clear directions to ensure that participants were able to use the checklist to perform the code review.

Observation 2: The error categories with the highest total percentage (> 60%) of error reported by students were: arrays/ linkedList/Trees, Output and flow of control. Initialization, declaration and method calls were the least reported error with 58% of the total errors being reported for each category by students. From Figure 2, we notice that students did not report any errors in the pilot study (Code 1) for the output format error category. Because this is the first time students were exposed to a checklist, students did not thoroughly conform to the process of using the checklist in the pilot study. Flow of control category was the only error category where students maintained a progressive percentage increase in reporting errors. We noticed a visible improvement in students' ability to find flow of control errors, which covers logical errors (something an integrated development environment (IDE) cannot find). It shows that students are able to critically analyze individual pieces of code from a logical perspective.

Observation 3: Based on the results in Figure 2, we also noticed that the performance of students across each pieces of code fluctuates for the majority of the error category with the exception of flow of control error type. This is because the complexity of each piece of code differs, which provided a different level of challenge for the students to recognize and report the errors. In addition, the topic being covered in each piece of code was progressively more challenging than the previous topic. There was a significant increase in the percentage of output format error reported for code 3 because students mentioned (in their reflections) that they seemed to have a better understanding of this particular topic area. This is interesting because it allows students to self-assess their knowledge based on their code review performance and can also guide instructors at identifying a good baseline knowledge at an individual and a class level. Flow of control error category maintained a steady increase in the overall percentage of errors

reported by students. Majority of the students who participated in the study have completed CS1 and are familiar with using an IDE. Some errors that would normally be caught by the compiler, students may have grown accustomed to having the IDE report those and therefore be less prone to identifying them manually. This lead to students progressively reporting more flow of control (logical) errors than they do others.

Observation 4 (Based on reflection sheet output): We analyzed student comments gathered from the reflection sheets to gain insight on the reasons students were unable to locate particular types of errors. Across each category of errors, more than 90% of the responses agree that they "*did not notice the error*"; in other words, they were unable to report those errors. Based on the survey questionnaire administered at the end of the guided PCR code sessions, we also discovered that some students thought the checklist needed clearer guidelines as some of students abandoned it mid-way and used an ad-hoc approach. This is not surprising but does inform how improvements can be made to the checklist. In the future, to help students make better use of the checklist, we will make the steps more specific, and provide training or a training document on guided PCR.

Research Question 2 (RQ2): *What is the effect of guided PCR on students learning of programming concepts in CS2 course?*

Observation 5 (Correlational analysis): We performed correlation analysis between the total number of category errors reported per topic during guided PCR and the students' performance on assignments covering specific topics. Based on the results in Table 2, students who are able to find more errors during the PCR session on a particular topic are able to score better on their lab assignments that covered the same topic. There was a positive (but non-significant, with the exception of Code Trees assignment) correlation across all four PCR sessions and their subsequent assignment performance. This suggests that students who are able to find more errors in someone else's code were able to retain and use that information when developing their own code. This also reinforces that the performance on the PCR session can be a good predictor of students' understanding of a particular topic.

Table 2: Correlation analysis between specific topics covered in guided PCR and specific assignment

Code 1 vs Code 1 lab Assignment	Code 2 vs Code 2 lab Assignment	Code 3 vs Code 3 lab Assignment	Code 4 vs Code 4 lab Assignment
r = 0.1427	r = 0.486	r = 0. 6381	r = 0.522
p = 0.5848	p = 0.047935	p = 0.005846	p = .031607

Research Question 3 (RQ3): *What perceptions do students have on guided PCR and using it in CS2 course?*

At the end of the four week guided PCR sessions, a feedback survey was conducted. The goal of the survey was to gain insights on how useful the error checklist was in PCR and how impactful it was on students' understanding of programming. Students rated their responses on a 5-point Likert scale [1= Strongly Disagree; 2= Disagree; 3= Neither Agree nor Disagree; 4= Agree; 5= Strongly Agree]. On examining the survey (see Figure 3), we found that over 68% of the

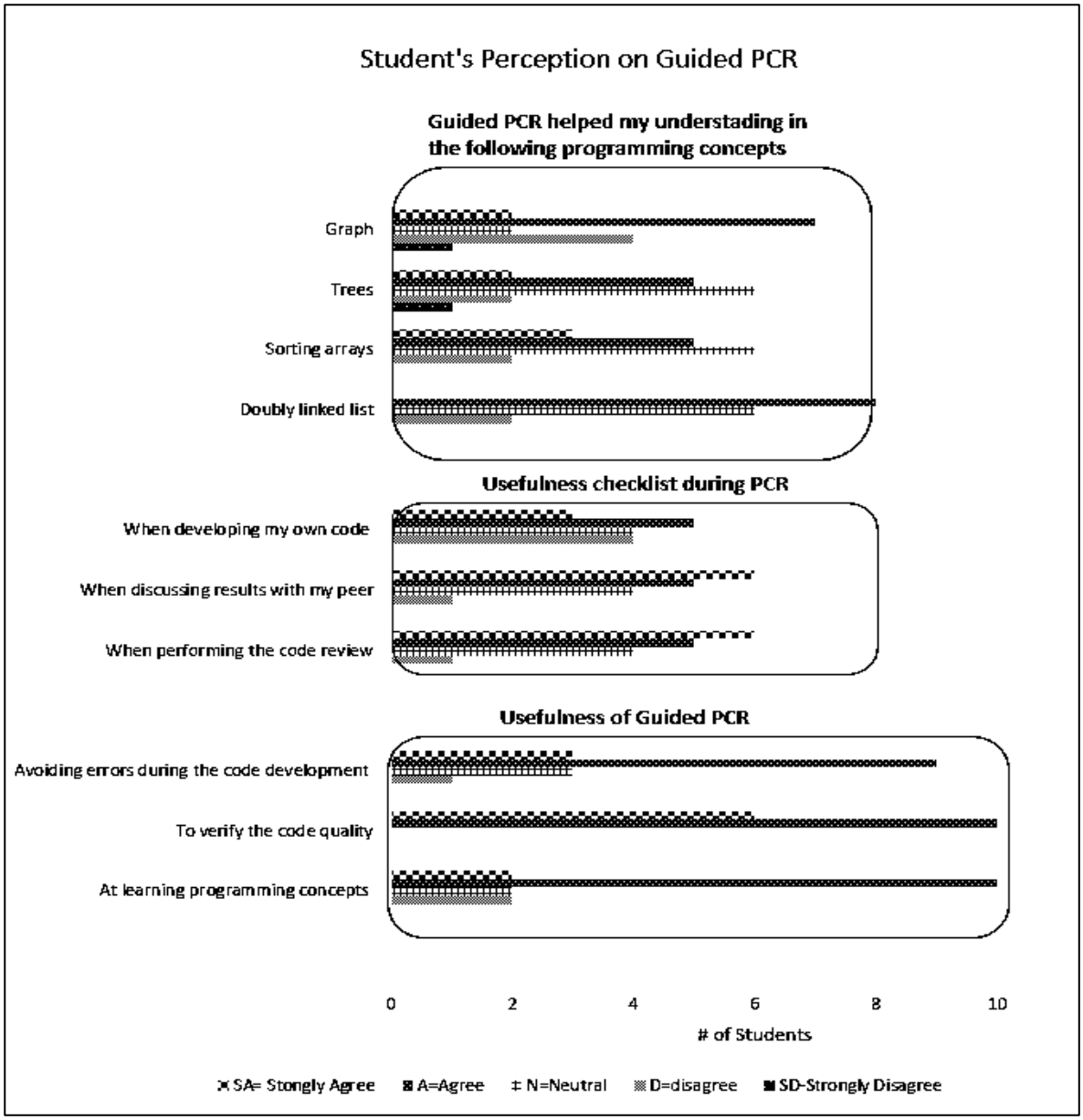

Figure 3: Survey results on Students' Perception of PCR

respondents indicated that PCR assisted in avoiding errors during their own code development, and 75% of respondents agreed that PCR helps in learning programming concepts. When asked if PCR is useful in verifying code quality, 100% of the students unanimously agreed. There was

positive feedback on the use of checklists in PCR with respect to how students felt about its application when developing their own code, when discussing results with peers, and when performing individual code reviews. Most of the students strongly agree that checklist was useful when performing PCR. Figure 3 shows the topic areas students were able to improve their understanding of programming concepts when using guided PCR. Overall, we saw an improvement in students understanding CS2 programming concepts, specifically in the areas of graph searching, trees traversal, and array sorting. Survey results show that over 80% of the students would recommend using PCR in CS2 classroom.

5. Discussion

We performed independent t-test analysis on two separate CS2 course. One course was administered in summer 2018 and the other in summer 2019. The same instructor taught both. The purpose of the test was to determine if with guided PCR students receive higher grades on their final exam (Summer 2019) versus the traditional teaching method that was implemented in the CS2 course in the previous years (summer 2018). The result shows that students in guided PCR achieve higher grades (The p-value = 0.016711) on their final exams compared to the previous years (when PCR was not used). There was an overall 21.666% percentage increase in students' final exam for guided PRC compared to a CS2 course that did not participate in the experiment. While not all of this may be attributable to the use of guided PCR, the improvement in the performance is noteworthy and warrants further investigation in a controlled study.

We also conducted a group review session and analyzed the new errors that were reported during the group review step of the PCR session. We tabulated the average number of errors reported by students during the individual review vs. the average number of new errors reported during the group review (when students were paired). This was done for all four guided PCR sessions and the results are shown in Table 3. Based on these results, students though may be able to exchange their individual error performance with each other, are not able to find a large number of new errors (that none of the pair members had previously discovered).

Table 3: Individual vs Group Errors Found

	Code 1	Code 2	Code 3	Code 4
Individual	23	78	72	72
Group	13	10	3	3

At the end of the summer 2019 survey, we asked students to provide additional comments. The comments provided by students gave us additional insights on usefulness of guided PCR in CS2 course. Below are some of the comments students provided followed by insights:

A. ***Students' feedback:*** *Group Discussion helps understand how each piece of code functioned.*
Insights: Students mentioned that they were able to think critically on each piece of code when working in pairs because they were able to discuss their perspectives and share

knowledge. While this may not have resulted in finding a large number of additional errors, students enjoyed discussing their individual performances with their peers. This activity also helped students at discussing different things to look for when reviewing code and ways in which they can solve data structures and algorithms problems.

B. ***Students' feedback:*** *Guided PCR was more helpful when code was more complex in design*
Insights: This exercise exposes students to the experience of performing code review irrespective of the code complexity. Students are able to see the benefit of using a checklist when they are given a more complex program to review versus a less complex code. This experience student can take with them to the software industry. Additionally, instructors can use this feedback from students to determine more difficult to comprehend topics that they should target when using guided PCR in CS2 course.

C. ***Students' feedback:*** *Error checklist needs improvement*
Insights: We discovered that some students were unable to follow the error checklist thoroughly because steps were unclear and some of the students found the checklist distracting. The researcher plan to make improvements on the error checklist prior to conducting further experiment. We will also need to conduct more extensive training that will communicate to the students the purpose of the checklist in PCR and demonstrate to the students the correct way to use the checklist.

D. ***Students' feedback:*** *It would be good to keep doing it as it helped us make fewer errors when writing own code*
Insights: Students find guided PCR exercise valuable in teaching CS2 programming concepts. Students felt that they could benefit from participating in additional PCR sessions because they are able to self-assess their understanding of computer programs when reviewing their performance against the true list of seeded errors. This when done over a longer period of time can help students become better programmer as they are more mindful of mistakes they have discovered previously.

E. ***Students' feedback:*** *Incorporate more CS2 topics*
Insights: Students commented that PCR could have benefitted their understanding of other CS2 topics. Some of these include teaching the concepts of static variable and polymorphism. This shows that students are experiencing difficulty in understanding those mentioned programming concepts and guided PCR can be used in the future to improve students' understanding of these concepts.

6. Conclusion

In this paper, we described how we utilized guided PCR to assist in students' knowledge acquisition of programming concepts in CS2 course. We have provided empirical data to support the increase in students' performance on assessments post guided PCR. The study shows that the current research was able to achieve the following:

- Assist students in learning data structure and algorithms programming concepts
- Improve students' performance on lab assignments and final exam

- Facilitate communication among peers

Threat to Validity: Factors that may have affected the results of the research includes-: The relatively small number of participating subjects, which limits the impact of the results. We plan to conduct additional studies to address this threat. Second, the defects seeded in the code snippets were done manually may not be representative of naturally occurring faults. We plan to conduct further studies to address these threats.

Future Work: We plan to improve on the current error checklist and incorporate a training process that covers the following:

- Convey to the students the importance of guided PCR
- Demonstrate to the students the proper way of using guided PCR

We also plan to conduct further investigation on the usefulness of guided PCR in a controlled study.

Overall students had a positive experience using guided PCR and recommended frequent use of this exercise in CS2 course by instructors.

7. Acknowledgement

This work is supported in part by the National Science Foundation under grant DUE-1525414. Any opinions, finding, and conclusions or recommendations expressed in this material are those of author(s) and do not necessarily reflect the views of the National Science Foundation.

References

[1] T.Baum, O. Liskin, K. Niklas, and K. Schneider (2016). "A Faceted Classification Scheme for Change-Based Industrial Code Review Processes". Software Quality, Reliability and Security (QRS), 2016 IEEE International Conference on, Proceedings. doi:10.1109/QRS.2016.19

[2] Y. Wang, H. Li, Y. Feng, Y. Jiang, Y. Liu, (2012). "Assessment of programming language learning based on peer code review model: Implementation and experience report, Computers & Education, Volume 59, Issue 2,2012,Pages 412-422,ISSN 0360-1315, https://doi.org/10.1016/j.compedu.2012.01.007.

[3] E.F. Gehringer, D.D. Chinn, M.A. Pérez-Quiñones, M.A. Ardis (2005) "Using peer review in teaching computing"SIGCSE '05 Proceedings of the 36th SIGCSE technical symposium on computer science education, ACM New York, NY, USA, St. Louis, MO, USA (2005), pp. 321-322

[4] Bloom, Benjamin S. (editor), Taxonomy of Educational Objectives: Handbook 1: Cognitive Domain. New York, Longman, 1956.

[5] B. Rosenshine (2012) "Principles of Instruction: Research-Based Strategies That All Teachers Should Know", ; American Educator Vol. 36, No. 1, Spring 2012, AFT

[6] M.V. Mäntylä, C. Lassenius (2009) “What types of defects are really discovered in code reviews”?IEEE Transactions on Software Engineering, 35 (3) (2009), pp. 430-448

[7] A.D.D. Cunha, D. Greathead “Does personality matter?: an analysis of code-review ability” Communications of the ACM, 50 (5) (2007), pp. 109-112

[8] D. A. Trytten. 2005. A Design for Team Peer Code Review. Proceedings of the 36th SigCSE Technical Symposium on Computer Science Education. Feb 2005. v.37 n.1.

[9] Li, X. (2007). Incorporating a code review process into the assessment. In the 20th annual conference of the National Advisory Committee on Computing Qualifications (NACCQ 2007) (pp. 125–131). Nelson, New Zealand.

[10] D.A. Trytten. 2005. A design for team peer code review. SIGCSE '05: Proceedings of the 36th SIGCSE technical symposium on Computer science education

[11] W.Yan-qing, L. Yi-jun, M. Collins and L. Pei-jie. 2008. Process improvement of peer code review and behavior analysis of its participants. SIGCSE '08: Proceedings of the 39th SIGCSE technical symposium on Computer science education

[12] B.M. Teasley (1993) Program Comprehension Skills and Their Acquisition: A Call for an Ecological Paradigm. In: Lemut E., du Boulay B., Dettori G. (eds) Cognitive Models and Intelligent Environments for Learning Programming. NATO ASI Series (Series F: Computer and Systems Sciences), vol 111. Springer, Berlin, Heidelberg

[13] A. Kolawa and D. Huizinga (2007). Automated Defect Prevention: Best Practices in Software Management. Wiley-IEEE Computer Society Press. p. 260. ISBN 0-470-04212-5.

[14] Brown, T.*, Reddy, M.*, Singh, M., Walia, G. “Using Code Review to Improve Programming Skills of Students in an Introductory CS Course”, IEEE Conference on Frontiers in Education, FIE-2019. October 16-19, 2019, Cincinnati, OH, USA