The Wisdom of Peers: A Motive for Exploring Peer Code Review in the Classroom

by

Mike Conley

A thesis submitted in conformity with the requirements
for the degree of Master of Science
Graduate Department of Computer Science
University of Toronto

Abstract

The Wisdom of Peers: A Motive for Exploring Peer Code Review in the Classroom

Mike Conley
Master of Science
Graduate Department of Computer Science
University of Toronto
2011

Peer code review is commonly used in the software development industry to identify and fix problems during the development process. An additional benefit is that it seems to help spread knowledge and expertise around the team conducting the review. So is it possible to leverage peer code review as a learning tool? Our experiment results show that peer code review seems to cause a performance boost in students. They also show that the average total peer mark generated by students seems to be similar to the total mark that a graduate-level teaching assistant might give. We found that students agree that peer code review teaches them something – however, we also found they do not enjoy grading their peers' work. We are encouraged by these results, and feel that they are a strong motive for further research in this area.

Acknowledgements

This work would not have been possible without the help of a number of people.

I would like to thank my second reader, Yuri Takhteyev, for helping me put on that final layer of polish. A great many ideas were bounced off of my fellow computing graduate students: Zuzel Vera Pacheco, Alecia Fowler, Andrew Smith, Jon Pipitone, and Alicia Grubb. Karen Reid, Diane Horton, Paul Gries, and Dr. Jim Clarke were instrumental in the creation of my assignment specifications and rubrics. Alan Rosenthal was invaluable for all of his technical support. Thanks to all of my experiment subjects, and my two anonymous graders.

I would also like to thank my friends and my family – in particular, my parents Brian and Sylvia. Thank you and much love to Emily Derr for keeping me focused, and listening to me ramble about this research! Also thank you to the entire extended Derr clan for all of the support. A special shoutout goes to Doug McQuiggan for all of his help with the initial experiment design, and his advice on data analysis.

Finally, I should point out that absolutely *none* of this would have been possible without the support and guidance of my supervisor, Dr. Gregory Wilson.

So, to all of you, thanks. And if I ever talked about my research to you, and you are not listed, a thanks to you, too!

This work is dedicated to all of you.

Contents

Chapter 1

Introduction

Peer code review is commonly used in the software development industry to identify and fix problems during the development process [23, 6]. Besides finding software problems, peer code review also has the added side-effect of spreading knowledge, expertise, and development techniques around the team involved in the review [23, 6, 28, 7].

Beyond being mentioned in a few slides in software engineering courses, however, peer code review is seldom part of the undergraduate computer science experience. Indeed, we find peer code review conspicuous by its very absence, especially considering its known learning benefits, and its prevalence in the software development industry.

Having spoken with several undergraduate computer science instructors at the University of Toronto, we believe that peer code review is not taught because:

1. Instructors already have difficulty with the amount of material in their teaching curriculum, and adding further material is not a trivial task for them.

2. Instructors are unsure of the pedagogical benefits of teaching peer code review.

3. Instructors are unsure of *how* to teach peer code review.

The second point here is critical – we must show that there is some pedagogical value in teaching peer code review. This is the primary concern of our research. We would also like to suggest that the third point can be addressed by inverting it: instead of trying to teach peer code review, perhaps we should be trying to use peer code review to teach. In doing so, we will have introduced the concept to students, and perhaps also found a way of teaching it.

As [13] and [10] both note, there has been a surge of interest in peer assessment within the past few decades, across a multitude of disciplines. However, despite several attempts (see: [22], [24], [13], [14], [26], [23]), quantitative research measuring the pedagogical effects of integrating peer code review into computer programming classrooms is still quite meagre.

For this research paper, we performed an experiment to measure the effects of code review on computer science students. In Chapter 2, we briefly discuss the theoretical background of code review and peer assessment. Towards the end of that section, we also introduce our research questions. In Chapter 3, we describe our experiment design and process in depth. In Chapter 4, we describe our technique and rationale for our data analysis. We present our results in Chapter 5. Chapter 6 illustrates several threats to validity that we discovered during our process. Chapter 7 lists several ideas and directions for future research. Finally, we finish with Chapter 8 which summarizes the conclusions of our research.

Chapter 2

Background and research questions

In this chapter, we will briefly survey some of the research concerning code review, then highlight some efforts by other researchers to introduce code review and peer assessments into the classroom. We will also use this chapter to introduce our research questions.

2.1 Code review in industry

In 1988, Boehm and Papaccio showed that the cost of fixing software defects is much smaller (by factors of 50 to 200) in the earlier stages of development than in the later stages [3]. There are many techniques in the software industry for detecting these defects early. One of those techniques is peer code review.

Peer code review as a software development process was first formalized by Michael Fagan in 1976 with what are now known as Fagan inspections. A Fagan inspection is a rigorous code review that takes place during an in-person meeting, with participants having predefined roles (Moderator, Designer, Author, Tester), and events conforming to a predefined sequence of steps (Preparation, Inspection, Rework, and Follow-up).

In [7], Fagan showed that his code review process dramatically reduced the number of defects injected into software during development. This discovery was later echoed by Wiegers in [28] and Cohen in [6]. Since Fagan's study, code review (in various forms) has been successfully used in industry to reduce the defects in software [28]. Cohen's study is particularly important, in that it shows that in-person meetings are not significantly superior to online, asynchronous code review sessions [6].

Along with defect reduction, a side-effect of the code review process seems to be the distribution of knowledge around the participants of the review. Fagan described this as giving inspection participants a high degree of product knowledge in a short time [7]. Similarly, Wiegers mentioned "additional benefits" of code review that are difficult to quantify [28]. He wrote that code review spreads product, project, and technical knowledge among team members, which helps the team establish shared expectations, reveals assumptions, and creates a shared understanding of the technical work products [28].

Cohen offered a similar sentiment in [6]. Cohen's contribution suggests that the learning potential of code reviews is not necessarily coupled to the in-person meetings advocated by Fagan and Wiegers. Instead, the learning comes from the individual, private inspection of code written by peers, and the ensuing public conversations about that code during the electronic review.

A key notion here is that it is not just the developers receiving the reviews who are learning from the reviewing process. It is also the developers formulating and giving the reviews. This is an important distinction which we will now discuss in the context of computer programming courses.

2.2 One-way code review in computer programming courses

In most computer programming courses, a form of "one-way" code review exists: the student writes code, the instructor or a teaching assistant grade (and sometimes annotates) the code, and the code is then returned to the student. The student can then react to the feedback they received on their next programming assignment or exercise. If Fagan, Wiegers and Cohen are correct regarding learning effects, then students following this passive, one-way model are only getting one-half of the benefits of code review, and are missing out on the potential learning gained by reviewing code themselves.

But, aside from solely taking the word of Cohen, Wieger and Fagan, is there any quantitative evidence showing that there are learning effects from giving code review? To explore this question, we reviewed several studies from a variety of disciplines with respect to peer assessment, and its effects on students. Our findings are described in the next section.

2.3 The learning benefits of peer assessment

Peer assessment as a learning tool and teaching strategy has been tried across a multitude of disciplines, by a variety of researchers. Work by Topping [21] and Falchikov [10] do a fantastic job of bringing all of these studies together, weighing their strengths, weaknesses, and claims. However, as both Sadler [20] and Turner [24] readily point out, there is a definite lack of quantitative studies to measure the effect on peer assessment on the assessors.

Still, this does not stop many of these researchers from making more qualitative, anecdotal, and speculative claims regarding the learning effects of assessing. Turner [24] and Sadler [20] both note that the act of assessing involves synthesis, analysis, and evaluation, which are all higher levels of Bloom's taxonomy [1]. Topping [21], Sadler [20], Falchikov([8] and [10]), and Roberts [19] argue that assessment is an incredibly deep learning experience, and is an opportunity for students to deepen and expand their knowledge of the subject. A number of researchers ([9], [19], [16], [2], and [25]) all praise the opportunity to develop critical thinking and analytical skills. Topping [21] and Roberts [19] highlight learning with regards to diplomacy and communication abilities. Roberts [19] and Venables [25] mention that peer assessment helps develop a sense of community and cooperative learning. Venables [25] and Gehringer [13] also highlight the value in students seeing the variety of styles, ideas, and problem-solving approaches of their peers.

The quantitative data to support these claims is difficult to obtain. Many of these studies focus instead on the performance of the students receiving the assessments, or do not make a distinction between those giving and those receiving assessments. Falchikov [9] briefly mentions a study that recorded better performance by peer assessors than students who did not participate in the study. Sadler [20], on the other hand, found that peer evaluators gained little from their assessment experience.

A recent Computer Science Ph.D. dissertation by Turner [22] presents what seems to be the most comprehensive study on assessor performance impacts, and his study conveniently centers itself on code review in computer programming courses. In [22], Turner discovered that peer code review is a good way of teaching the object-oriented programming concept of decomposition to reviewers. Furthermore, Turner found that students who performed peer code review were less likely to complete their assignments than students who did not perform peer code review. It should be noted that in [22], Turner et al. restricted themselves to measuring performance gains along a very limited set of criteria concerned mainly with high-level object-oriented principles. We find this restriction somewhat limiting, and feel that there is more room for exploration in this area.

This leads us to formulate our first research question:

RQ1 : Does the act of *performing* code review cause students to become better programmers? If so, in what ways do they become better?

2.4 The validity of peer assessments

The amount of research examining the validity of peer assessments is immense. Falchikov [10] and Topping [21] were primarily concerned with this topic. After surveying a multitude of studies that measured the validity of peer marks, Topping concluded that peer assessment is of adequate reliability and validity in a variety of applications [21]. In [10], Falchikov concurs, saying that she found definitive evidence of agreement between peer and teacher marks on average. Similar claims can be found in [9], [20], [17], and [2].

Whether or not this pattern holds in the context of computer programming courses is a different matter. Once again, the research here is quite thin. Wang et al. attempted to integrate code review into a course in [26], but had difficulty in organizing the process. Hundhausen et al. had students do a version of Fagan's formal inspections in their course in [14], and found that the number of found defects dropped dramatically over time. Despite these results, we had difficulty in finding a clear, quantitative study that measured the validity of peer code reviewer grades.

This leads us to our second research question:

RQ2 : To what degree do peer graders agree with graduate-level teaching assistants with respect to assigning marks to assignment submissions?

2.5 Student opinion of peer assessment

Student opinion of peer assessment is another heavily researched area, and is explored by a number of researchers (see: [27], [2], [19], [25], [9], [8], [20], and [4]). It is no surprise that there is so much attention given to this topic: it is important to have students "on board" with the idea of peer assessment in order for it to be effectively integrated into their learning.

Within these studies, students generally responded to the initial idea of peer assessment negatively, or with some skepticism. Numerous researchers (see: [27], [25], [9], [19], and [8]) found that at first, students felt unsure of their grading abilities, and did not trust themselves to do a good enough job. Similarly, several researchers (see: [27], [2], [4], and [19]) found that students did not trust the other students who would assess them to take the job seriously, and do a fair job. In [4], Cartney found that some students were embarrassed to show their work to their peers. Finally, several researchers (see: [27], [20], [25], and [8]) found that some students felt that assessment was an inappropriate task for them, and that it was really the place of the teacher to perform assessments. A more in-depth discussion on initial student reservations to peer assessment can be found in [4].

Interestingly, student opinion after experiencing peer assessment was almost always positive. Positive experiences by students were reported by many researchers (see: [13], [12], [16], [2], [4], [25], [9], [20], and [21]). In general, students felt that peer assessment was demanding, but a positive experience, reports Topping [21]. Topping in [21] and Cartney in [4] were among the few to present or cite studies where students were still skeptical of the peer assessment exercise after experiencing it.

With regard to peer code review in computer programming courses, both positive and negative student opinion is reported by Turner in [22]. In [14], Hundhausen et al. reported that a positive sense of community developed within their code inspection groups. Wang et al. found that the students took advantage of the loose code review structure in order to game the system in [26].

Clearly, there is more room for exploring the question of student opinion of peer code review in the computer programming course context. This leads us to our final research question:

RQ3 : How do students in computer programming courses feel about peer code review?

2.6 Summary

Our research aims to answer the following research questions:

RQ1 : Does the act of *performing* code review cause students to become better programmers? If so, in what ways do they become better?

RQ2 : To what degree do peer graders agree with graduate-level teaching assistants with respect to assigning marks to assignment submissions?

RQ3 : How do students in computer programming courses feel about peer code review?

We have designed an experiment to answer these questions. The following chapters will illustrate the experiment design and results, and also discuss the potential uses of what we have discovered.

Chapter 3

Method

In this chapter, we will describe our experiment method and materials. We start by defining some terms that we will use for the duration of the work. We then give an overview of the process of our experiment, followed by a brief description of our experiment participants. We conclude this chapter by describing the instruments that we used in our experiment.

3.1 Definitions

The following terms will be used in this work:

- **Assignment specification** : An explicit set of requirements to be satisfied for an assignment.
- **Grader** : A graduate level study participant who marked submissions created by the experiment subjects.
- **Marking** : Evaluating a submission based upon a predetermined marking rubric.
- **Marking rubric** : A scoring tool to facilitate marking of submissions. Marking rubrics have a set of criteria, and each criterion can be given a mark level from 0

to 4. Each level is given a description appropriate for the context of the criterion.

- **Peer grader** : A subject who marked submissions that we provided.
- **Peer grader average** : The average value that peer graders supplied. This might be for a particular rubric criterion, or the total mark for a submission.
- **Subject** : An undergraduate level study participant who worked on creating submissions for the assignment specifications.
- **Submission** : Material generated in order to satisfy an assignment specification.

3.2 Experiment design overview

Subjects were individually brought into the lab to create submissions for two assignment specifications. Before arrival, subjects were randomly assigned to one of two groups: treatment and control. The treatment group performed peer grading, and the control group did not. Submissions written by all subjects were then marked separately by two graders. This design allowed us to observe whether or not there was a significant change in the average marks between the control and the treatment groups, as well as giving us information on grader/peer grader marking agreement. An exit questionnaire for the treatment group was used to gather opinion data on their peer grading experience.

3.3 Writing sessions

Members of both groups filled out a background questionnaire to verify the required experience level of the subject. Subjects were then told that they had a maximum of half an hour to work on the first assignment specification, and that their work might or might not be seen by other subjects in the study. They were told that they had full usage of the Internet for help, and could ask for clarification if something in the assignment specification was unclear. A countdown application was started on the computer desktop to help the subjects track their time. After completing the first assignment specification, subjects spent half an hour performing another activity, determined by the particular group that they belonged to. The treatment group spent the time marking five mocked up submissions for the assignment specification that they themselves had just written, and the control group worked on completing a set of vocabulary exercises. Once this activity was completed, or the half hour time limit was reached, subjects in both groups had another half an hour to work on the second assignment specification. Subjects in the treatment group were subsequently asked to fill out a brief questionnaire to get feedback on their marking experience, and then members of both groups were released.

3.4 Grading sessions

Submissions from both groups were printed out on paper, mixed with the mock-ups, randomized, and then handed off to the graders for marking. Graders were kept unaware of the purpose of the study, and simply asked to mark the submissions as if they were for two assignments from an undergraduate course.

In an attempt to measure any changes in the code that the rubric did not account for, the next step gathered more general preference feedback from the graders. After marking was complete, the mock-ups were removed, and submissions for both assignments were paired up by author, and then randomized. Graders were told that the submissions had been paired up by author, and then were asked to go through each pair and choose which of the two submissions they preferred, and to give feedback on how different the quality of the two submissions were. Graders were not aware from which group the submission authors came from, nor which submissions were completed first and second. Feedback was provided on a grader preference form, which is described in greater detail in Section 3.6.8.

3.5 Subjects

Subjects for this study were recruited from University of Toronto undergraduate students. The only qualification for participation was four or more months of experience working with the Python programming language. Subjects were recruited by posting fliers in common public areas in the computer science department, posting messages on the department electronic bulletin board, and by announcements given during the beginning of courses. The incentive for participation was a prize draw for a $100 Best Buy gift card.

There were a total of thirty subjects. Before arrival, each subject was randomly assigned to either the control or treatment group, such that there were fifteen subjects assigned to each group by the end of the experiment sessions.

The two graders were graduate level students recruited through word of mouth, and were paid standard teaching-assistant wage for the time spent grading and evaluating the submissions.

3.6 Instruments

In this section, we will describe the instruments we used in our experiment.

3.6.1 Pre-experiment questionnaire

The pre-experiment questionnaire was an online form that asked subjects to provide the number of post-secondary months spent in a programming intensive job or course. We used this as a rough measure of how experienced a programmer the subject was, in order to ensure that they had the requisite skills to complete the experiment tasks. The pre-experiment questionnaire also asked the subject if they wanted to receive the results of the study upon completion, and allowed them to enter their e-mail address if they did.

3.6.2 Assignment specifications

Assignment specifications were developed in cooperation with several undergraduate instructors from the University of Toronto Department of Computer Science. These instructors gave advice and feedback on the type of assignment specifications usually given in undergraduate courses. Each specification focused on a different programming problem, but considerable effort was put into making both assignments equally difficult. To control for even minute differences in difficulty, the order that subjects worked on the assignment specifications was counterbalanced – subjects in each group would randomly alternate between which assignment was written first, and which assignment was written second. Each assignment specification was printed out and provided to the subjects on paper. The assignment specifications were:

Flights and Passengers : Flights and Passengers models passengers flying on an airline. Subjects had to implement five methods of a Flight class and work with a pre-compiled Passenger class. The Flight class stores a collection of Passengers, and must be able to represent the class as a string. The class must also be able to return the heaviest and lightest passengers in the collection based on their luggage weights. In order to complete a submission for this specification, subjects must understand classes, collections, and sorting / searching collections of collections.

Decks and Cards : Decks and Cards models a deck of ordinary playing cards. Subjects had to implement five methods of a Deck class and work with a pre-compiled Card class. The Deck class stores a collection of Cards, and must be able to represent the class as a string. The class must also be able to shuffle, deal, and cut the cards. In order to complete a submission for this specification, subjects must understand classes, collections, and be able to use and manipulate Python Lists.

3.6.3 Assignment rubrics

Rubrics for each assignment were developed in cooperation with the same undergraduate instructors who helped to develop the assignment specifications. Despite the differences between assignment specifications, the rubrics for each assignment were similar. Each rubric had fourteen criteria. Five criteria were for the correctness of each class method, and another five were for the design of each class method. There were an additional four criteria for overall style, docstrings, internal commenting and error handling. The rubrics were printed on paper, and were completed by filling in the appropriate number per criterion.

3.6.4 Mock-up submissions

The five submissions that the peer graders were asked to mark were mocked up by the investigators. There were five mock-ups for each assignment (ten mock-ups in total). Like the assignment specifications and rubrics, these mock-ups were created based on feedback from undergraduate instructors from the Department of Computer Science at the University of Toronto. This allowed for a relatively high degree of realism in the code quality of the mocked up submissions. Great effort was taken to ensure that the quality of mock-ups was similar across both assignments. The mocked up submissions were printed out on paper and provided to subjects to mark. Subjects were allowed to make notes on the code during their marking.

3.6.5 Vocabulary exercise

In order to control for fatigue from the half-hour time period that the treatment group spent marking the mock-ups, subjects within the control group were asked to work on five online vocabulary exercises and to self-report their scores. This task was presented in such a way that control group subjects believed that their full focus was required. Thus, subjects in the control group were actively engaged in a mental exercise for approximately the same duration that the treatment group spent marking. Like peer graders that completed marking quickly, subjects in the control group that completed all vocabulary exercises within the half hour limit were allowed to move on to the second assignment specification immediately. The exercises centred on vocabulary found on the SAT, TOEFL, GRE, and GMAT tests, as well as in the VOA Special English controlled version of the English language.

3.6.6 Post-experiment questionnaire

The post-experiment questionnaire was written by subjects within the treatment group, and consisted of an online form with several Likert scales designed to gather opinion data on the peer grading experience. For some questions, subjects were asked to provide additional comments if their opinion rating was three or greater on the one-to-five Likert scale.

3.6.7 Working environment

In order to minimize any unfamiliarity or discomfort with the working environment, subjects were asked for their preferred operating system and Python editing environment, and a computer meeting these specifications was prepared for them prior to arrival[1].

A quiet, well-lit office was used to host the study. Subjects had their own desk, and access to pencils and paper for organizing their thoughts. One researcher was available to guide the subject through the tasks, and to give clarification on the assignment specifications and rubrics.

[1]The computer was a PC, with a 3Ghz Dual Pentium 4 processor and 1GB of RAM memory. The peripherals included a 17" flat-screen monitor, a wired optical mouse, and a standard 103 key keyboard. For operating systems, the computer dual-booted into Microsoft's Windows XP SP3 and Canonical's Ubuntu Linux 8.04.

3.6.8 Grader preference form

In order to collect grader preference feedback on submissions paired by author, a web form was used. The web form asked the grader to choose which submission they preferred from each author—the submission for the Flights and Passengers assignment, or for the Decks and Cards assignment. The grader was then asked to rate how different the two submissions were on a scale from one to five, with one being not different, and five being very different. If graders answered with a four or a five on the scale, they were asked to describe in more detail what the difference was between both submissions.

Chapter 4

Analysis

In this chapter, we will describe how we analyzed our data and explain the reasoning behind our analysis decisions.

4.1 Agreement

In this section, we will narrow our definition of mark agreement, and introduce the notion of general agreement.

4.1.1 Precise agreement

Even with the structure of the marking rubric as a guideline, the evaluation of programming assignments such as the ones used in this experiment is a subjective activity. As such, it is common for different graders to give different marks to the same work. The problem of consistency, reliability, and validity in subjective marking has been heavily documented (see [9], [10] and [12] for example).

Prof. Jim Clarke, Senior Lecturer and Associate Chair for Undergraduate Studies in the University of Toronto Department of Computer Science, remarks:

> Of course they [disagree]. That's why we either split the marking by assign-

> ment or adjust [marks] between tutorials. Precise agreement would be pretty rare even if two graders used about the same standards [5].

However, while it is unlikely to have precise agreement from one grader to the next, it is possible that graders may still rank the quality of submissions in more or less the same fashion. We will call this similarity in quality ranking the general agreement between graders.

4.1.2 Measuring and comparing general agreement

General agreement is determined by measuring Pearson's Correlation Coefficient across independent samples of marks. This is the measurement technique used by [17], [20], and the majority of the studies examined by [10].

We used the marks for the mock-up submissions as the samples for our measurement of agreement, since both the treatment group subjects and the graders marked the mock-up submissions. We simply compared the peer average for each criterion with the marks for both graders. We also compared the peer average for the total submission mark with the same mark from both graders. The marks for each criterion were averaged over each mock-up, so that we had a single average mark per criterion, per assignment specification.

In order to compare the correlations between the graders and the peer average marks, we followed [17], and used Fisher's Z Transformation to measure the difference in correlations.

4.2 Computing the peer average

In [17] Joordens and Paré calculated their peer average of five marks by averaging across all but the highest and lowest mark [17]. We decided not to use this approach because Joordens and Paré only had to average a single mark, whereas we had all criteria for each assignment to average. We also felt that removing the highest and lowest mark per criterion per mock-up would add an unnecessary complication to our experiment, so we simply averaged across all values provided per criterion.

It should also be noted that in some cases, not all subjects were able to complete their peer grading on time. This means that some mock-ups received partial or no marks from those subjects. However, even in the worst case, each mock-up criterion had at least five values to average. We felt that this would be enough to calculate the peer average, and that any distortions caused by variance in the amount of values being averaged would be negligible.

4.3 Comparing subject performance in the control and treatment groups

In order to determine if the grading process had any effect on subject performance, the difference in marks for each criterion, from the first assignment to the second, was analyzed. This difference will henceforth be referred to as the mark gain on a particular criterion.

Since it is illogical to compare assignment-specific criteria across two different assignments, the marks for several criteria on each assignment had to be aggregated into a more general form in order to do any reasonable comparison. In particular, the assignment-specific design and correctness criteria were aggregated into a single design and correctness sum for each submission. When we later discuss the mark gains for design and correctness, we are referring to the difference in these aggregations between the first and second assignments. The total submission mark is also an aggregation that we measured for mark gain. The correctness and design criteria aggregates were then normalized, to scale with the rest of the individual criteria. Likewise, the total submission mark was normalized for each of the fourteen criteria of each assignment rubric

4.4 Measuring grader preference

The data gathered from the grader preference form was split based on the group that the submission author belonged to. For the data from both graders, the values from the Likert scale to indicate difference in quality were averaged for each group, and a Student's t-test was used to determine if the two averages were significantly different.

Next, we measured how often the graders chose the second assignment for both groups. Instances where the grader chose the second assignment, but also chose the lowest value on the Likert scale to indicate difference in quality, were not included. This allowed us to discard any false positives generated when the grader did not notice any difference between the two submissions, and chose the second assignment arbitrarily.

The proportions of times the second assignment was chosen in either group was compared using Fisher's Exact Test, due to the small sample size.

Chapter 5

Results

In this chapter, we will present and discuss the results of our experiment. A summary of our findings can be found at the end of this chapter.

5.1 Grader and peer average agreement

The purpose of this section is to show that the peer average marks were similar to the marks from both graders. In order to measure the difference between the peer average and the grader marks, we first measured the difference in agreement between the two graders in order to have something that we could compare against.

To determine grader agreement, we measured the correlation between the marks given by both graders. As mentioned in Section 4.1.2, we used an analysis methodology similar to Joordens and Paré in [17] by using Pearson's Correlation Coefficient to measure the grader agreement over each criterion for all ten mock-ups. We also measured agreement over the total submission mark per mock-up. The tables with those measurements can be found in the Appendix, Section 9.1.

We then compared the peer grader average calculated for each mock-up submission to both of the grader marks, to see if there were meaningful differences in agreement. As mentioned in Section 4.1.2, we followed Joordens and Paré's example, and used Fisher's

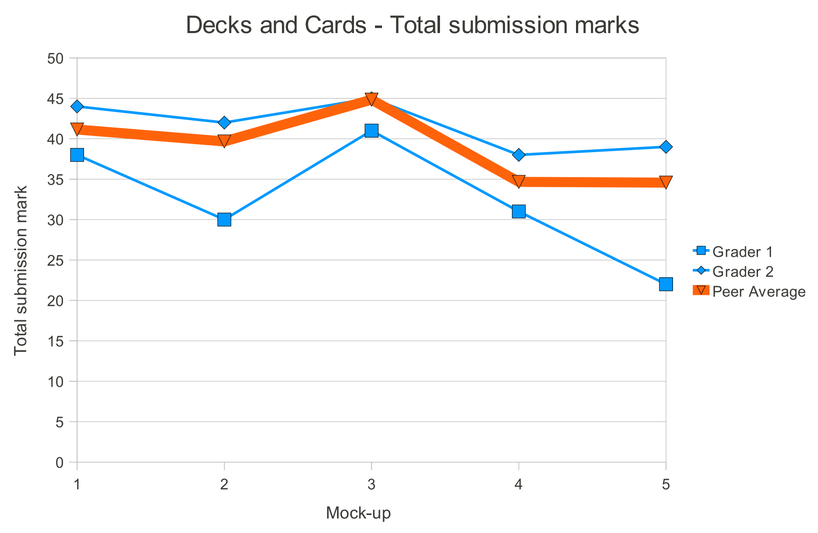

Figure 5.1: The total submission marks from both graders and the peer average for the Decks and Cards assignment

Z Transformation to compare the correlations for each assignment criterion, aggregates, and total submission mark. The tables with those measurements can be found in the Appendix, Section 9.2.

We found no conclusive evidence to show that the two graders agree with each other any more or any less than they agree with the peer average over each individual criterion. We did, however, find indications of a meaningful agreement between the graders and the peer average over the total submission mark ($p < 0.10, N = 5$). See Figures 5.1 and 5.2. Notice in particular how, in eight cases out of ten, the total submission mark given by the peer average fell between both graders.

This data supports the hypothesis that the peer average gives total submission mark feedback with a reliability and validity comparable to an expert grader.

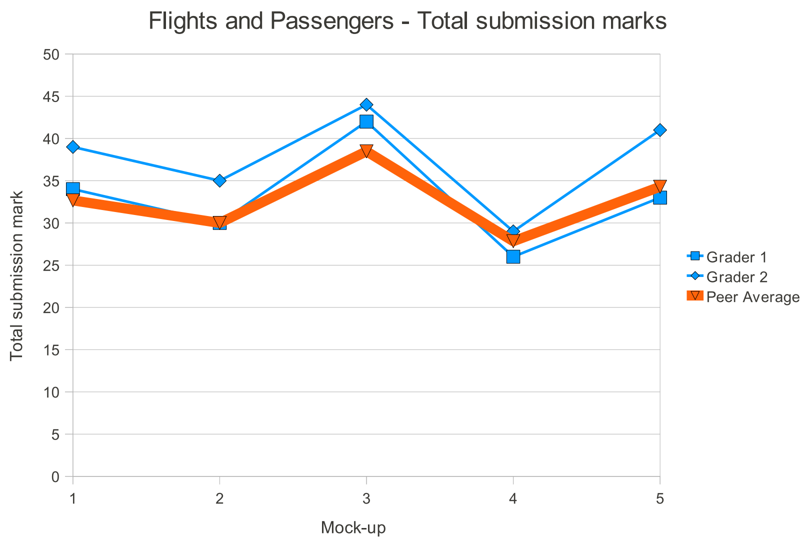

Figure 5.2: The total submission marks from both graders and the peer average for the Flights and Passengers assignment

5.2 Treatment vs. control

In this section, we will compare the treatment and control groups in terms of performance and grader preference. In particular, we will show that the grading process had a positive impact on performance for the treatment group, but that outside of using the marking rubric, graders could not seem to detect a difference between groups.

5.2.1 Performance

Our marking rubric allowed us to break our analysis down by criterion in our measurements of mark gains between groups from both graders. The means and standard deviations for the mark gains can be seen in Table 5.1.

As mentioned in Chapter 4, Section 4.3, the correctness and design criteria were aggregated, and then normalized. Likewise, the total submission mark was normalized for each of the 14 criteria of each assignment rubric.

Our data here shows that the treatment group had a larger improvement than the

Criterion	**Control**		**Treatment**		**p-value**
	MG Mean	**MG StDev**	**MG Mean**	**MG StDev**	
Correctness aggregate. (normalized)	0.09	0.59	0.43	0.82	0.21
Design aggregate (normalized)	-0.04	0.67	0.52	0.94	0.08
Error checking	0.1	2.29	0.37	1.91	0.75
Style	-0.1	0.81	0.5	1.02	0.08
Docstrings	0.07	1.41	0.73	1.75	0.28
Internal comments	0.00	1.25	0.67	1.03	0.07
Total submission mark (normalized)	0.02	0.48	0.5	0.71	0.05*

Table 5.1: Mark gain means and standard deviations for both groups, for criteria and criteria aggregates

control group in each of the seven criteria / aggregates, though in some of the areas the difference was not statistically significant. What is most interesting, is that the total submission mark shows a mark gain of 0.5 for the treatment group, over the 0.02 gain for the control group. Using Student's t-test, we have found a positive difference in the treatment group, with $p < 0.10, N = 15$ (see Table 5.1*).

This data supports the hypothesis that students who do peer grading will show performance gains over students who do not do peer grading.

5.2.2 Grader preference

We were curious to find out if there were performance gains in the treatment group that our rubric did not account for. Separate from the marking rubric data, we asked graders to simply choose which submissions they preferred the most, as described in Section 4.4.

We found no evidence to suggest that, outside of the marking rubric, graders preferred submissions from the treatment group any more than submissions from the control group. The data that we analyzed can be found in the Appendix, Section 9.3.

5.3 Subject opinion on peer grading

The responses from the post-experiment questionnaire were put into a frequency table, and can be seen in Table 5.2. We have also included the mean and standard deviation over the responses for each part of the questionnaire.

It should be noted that when the subjects are referring to their peers' work, they are really referring to the mock-ups that they had graded. This distinction is discussed further in Chapter 6.

We will now discuss what we think are the most interesting parts of Table 5.2.

5.3.1 Subjects tended to agree that seeing their peers' code taught them something, regardless of reported experience level

We asked our subjects to respond to the statement "Seeing my peers' code taught me things I didn't already know.". For this statement, we see a very clear tendency towards agreement. Subjects who answered 3 or higher were asked to expand on their answer by providing details on what they felt they had learned. Responses tended to fall into two categories:

Tricks and shortcuts : One respondent mentioned that the grading process taught them some tricks or shortcuts that made code seem more elegant or concise. Another mentioned that the grading had taught them how to randomize lists, and about a better way of dealing with program errors.

Statement	**# of responses**					**Mean**	**StDev**
	1—Strongly Disagree	**2**	**3**	**4**	**5—Strongly Agree**		
"It is unusual for me to see code written by my peers."	3	1	**5**	4	2	3.06	1.33
"Seeing my peers' code taught me things I didn't already know."	1	1	4	2	**7**	3.87	1.30
"Because I saw and graded my peers' work, I believe I know more about the quality of my own work."	0	1	1	6	**7**	4.27	0.88
"I am interested in knowing how my peers graded me."	0	1	1	4	**9**	4.4	0.91
"I would have written the code for my first assignment differently if I had seen the rubric beforehand."	1	4	2	3	**5**	3.47	1.41
"During this experiment, I enjoyed seeing other student's assignments."	3	0	1	**6**	5	3.67	1.50
"I enjoyed grading my peers' work."	3	3	**4**	**4**	1	2.80	1.26
"I found grading my peers' work difficult."	1	3	**5**	4	2	3.20	1.15
"I'm confident that the grading I did was fair."	0	0	**8**	7	0	3.47	0.52
"Because I knew that my peers would be seeing and grading my code for the first assignment, I coded it differently than I would have normally."	**7**	6	1	0	1	1.8	1.08

Table 5.2: Breakdown of treatment group responses for post-experiment questionnaire

For this group of respondents, the grading was teaching them practical lessons that could be applied to their software writing.

The way my peers think : The majority of comments fell into this category. Subjects in this category expressed that the grading taught them that there are sometimes many ways of reasoning about and solving problems, and that some ways are more clear and elegant than others.

We were curious if there was any relationship between the responses that subjects gave on this statement ("Seeing my peers' code taught me things I didn't already know."), and the number of months of experience they reported in the pre-experiment questionnaire. In particular, we wondered if subjects who reported less experience would be the ones who strongly agreed with the statement.

To analyze this, subjects were grouped by their response to the statement, and the average reported months of experience were compared. The results can be found in the Appendix, Section 9.4. We found no evidence that subjects who reported less experience agreed with the statement more. In fact, the subject who reported the highest experience level strongly agreed with the statement as well.

The conclusion we draw from this is that the subjects tend to say that peer grading teachers them something—even if that something is just another approach to the same problem. This seems to be true for subjects regardless of their reported experience level.

5.3.2 Subjects tended to agree that seeing their peers' work gives them a sense of the quality of their own work

We asked our subjects to respond to the statement: "Because I saw and graded my peers' work, I believe I know more about the quality of my own work." Once again, there is a clear tendency towards agreement. The subjects tend to say that they agree that the peer grading process is good at giving them a sense of how their own work rates in comparison with other work.

5.3.3 Subjects tended to agree that they are interested in knowing how they were graded by peers

We asked our subjects to respond to the statement: “I am interested in knowing how my peers graded me.” Here, we noticed another clear tendency towards agreement. The subjects tend to say that they value the opinion of their peers, and are very interested in knowing how their work is rated by others.

5.3.4 Subjects tended to agree that seeing the rubric would have affected their performance on the first assignment

We asked our subjects to respond to the statement: “I would have written the code for my first assignment differently if I had seen the rubric beforehand.” Though the majority of the weight appears to be on the agreement side, opinion seems to be split. This was another question where we asked our subjects to provide more detail if they answered 3 or above. The majority of these comments mention how the subjects would have put more effort into documentation and style had they reviewed the rubric beforehand. This could indicate that the assignment specification was unclear as to how the subjects were going to be evaluated.

5.3.5 Subjects tended to agree that they enjoyed seeing their peers' work

We asked our subjects to respond to the statement: “During this experiment, I enjoyed seeing other student's assignments.” Once again, most of the weight is on the agreement side. The subjects tend to say that they enjoyed the act of seeing and reading code written by their peers.

5.3.6 Subjects tended not to enjoy grading their peers' work

We asked our subjects to respond to the statement: "I enjoyed grading my peers' work." Here, the majority of the weight appears to be towards disagreement. It looks that, while subjects did enjoy the act of seeing other submissions, in general they did not seem to enjoy the process of grading the other submissions.

5.3.7 Some subjects found grading to be difficult

We asked our subjects to respond to the statement: "I found grading my peers' work difficult." Here, the responses are almost split exactly down the middle, with only a slight emphasis on the agreement side.

For this question, subjects who answered 3 or higher were asked to expand on their answer by providing details on what they felt was difficult about the grading process. Here are just a few of the comments we received:

- "The hardest part was trying to trace through messy code in order to figure out if it actually works."
- "It just became really tedious trying to understand people's code."
- "[It was difficult] [r]eading through convoluted, circuitous code to determine correctness."
- "[It was difficult] [b]eing harsh and honest. I guess it's good not to ever meet the people who wrote the codes [sic] (unlike TAs) because they aren't there to defend themselves."
- "Not every case is clear-cut, and sometimes it's hard to decide which score to give."
- "Giving bad marks are [sic] hard! Reading bad code is painful! It wasn't fun!"

- "Emotionally, I know what the student is doing but I have to give bad marks for comments or style which makes me feel bad. "

These subjects clearly did not enjoy grading the convoluted code we had provided in the mock-ups. It is also worth noting that some of these subjects felt badly about giving their peers poor grades.

Some subjects also used this opportunity to criticize the amount of time available for marking, and the strictness of the rubric. We acknowledge and comment on these complaints in Chapter 6.

5.3.8 Subjects tended to only be moderately sure that the grading they did was fair

We asked our subjects to respond to the statement: "I'm confident that the grading I did was fair." The answers for this question are in the neutral-agreement range. There is not a full agreement, but there is also no disagreement. This suggests to us that our subjects were not strongly confident in their ability to grade.

5.3.9 It is unclear if the knowledge that their peers would be grading them affected our subjects work

We asked our subjects to respond to the statement: "Because I knew that my peers would be seeing and grading my code for the first assignment, I coded it differently than I would have normally." It is tempting to examine this data and conclude that students would have written their assignments the same, regardless of whether or not they knew that their peers would be grading them. However, there was a flaw in our experiment methedology—subjects were aware that other subjects might see their work, but they were not told that other subjects might grade their work. That puts the significance of these answers under question. Despite the signal towards disagreement in the data, we have decided to toss out this data as inconclusive, and tainted by experimental error.

5.4 Summary of results

Our results can be summarized as follows:

- The peer grade average of the total submission mark generated by the treatment group seemed to be in agreement with the two graders.
- There was a small, yet significant gain in performance for students in the treatment group on their second assignment.

And with regard to subject opinion of the peer grading process, subjects tended to agree that:

- Seeing their peers' code taught them something new, regardless of their reported experience level.
- Seeing their peers' code gave them a better sense of their own code quality.
- They were interested in knowing how their peers graded them.
- They enjoyed the act of seeing and reading their peers' code.
- They believed that they graded fairly, but were also not 100% confident in their grading abilities.
- They did not enjoy the act of grading their peers' code.

Chapter 6

Threats to Validity

During the course of the experiment and data analysis, we discovered several experimental design concerns. We will discuss those concerns here.

6.1 Small Sample Size

Resource restrictions prevented us from performing our experiment on a large sample of subjects. Subsequently, our data set is quite small. Therefore, our claims should be treated with some level of scepticism. We feel it is a bit premature for us to attempt to draw definitive conclusions from our analysis, and so we have restricted ourselves to presenting only plausible hypotheses.

6.2 Priming / activation effects

The performance boost detected in Chapter 5, Section 5.2.1 should not be considered definitive evidence of learning. The priming effect has been clearly established and studied in the field of psychology (see [11] for more information on the priming effect), and we cannot rule out the possibility that the performance boost is a result of the priming effect.

6.3 Subjects were misled

The opinions of subjects presented in Chapter 5, Section 5.3 do not take into account the fact that the subjects were misled into thinking that they were marking their peers' work. The subjects were not told that the submissions that they were marking were mock-ups.

6.4 Acquiescence bias

The opinions of subjects presented in Chapter 5, Section 5.3 also do not take into account acquiescence bias – or the tendency for subjects to agree with our questions or indicate a positive connotation (see [15] for more information on acquiescence bias).

6.5 The mocked up submissions might not be an accurate representation

Despite the amount of care that was taken in the creation of the mock-ups to accurately represent the quality of work that instructors might receive from undergraduate students, it is still possible that these mock-ups are not an accurate enough portrayal. If this is true, then we cannot generalize our results with respect to code created by real undergraduate students.

6.6 There was insufficient time to perform grading

There were several instances where subjects in the treatment group were unable to complete the marking of all five mock-ups. There were also other instances where subjects completed the marking of all five mock-ups, but then expressed to the investigator that they doubted they did a careful enough job due to the time constraints. This throws the integrity of the peer average under some suspicion, as some subjects may have sacrificed marking quality for speed. This also could have affected subject opinion on the grading process.

6.7 The rigidity of the marking rubrics

The marking rubrics received the same level of careful design as the assignment specifications. However, some subjects mentioned that the rubrics failed to account for certain things. For example, one subject mentioned that the internal comments criterion on the rubric only concerned itself with the amount of useful comments, and did not take into account self-documenting code. For example, a submission that was composed entirely of self-documented code would be penalized on the internal comments criterion.

This suggests that our rubric may have been too rigid in its design, which could have affected the overall marking process for both graders and treatment group subjects.

6.8 External generalizability

All experiment subjects were drawn from the University of Toronto's Department of Computer Science. Therefore, if subject performance is related to course curriculum, then the results from this study might not generalize for students from other institutions.

Chapter 7

Future Work

We are encouraged by the results in Chapter 5, and we believe that there is motive for future research in this area. We will now outline just a few possible directions that future research could take.

7.1 Increasing the size of N

We would perform the same experiment, this time with more resources, and more subjects to gather data from. We would then analyze the data to see if the results are similar the results of this experiment. We would also hope to address some of the issues brought up in Chapter 6 – in particular, we would re-design the post-experiment questionnaire to account for acquiescence bias, and also give the subjects more time to grade.

7.2 Integrating code review into a live computer science course

We have shown that students might benefit from performing peer code review, and that these peer reviews themselves might generate marks that are just as reliable as graduate-level teaching assistants. This is very much in line with the research that Joordens and Paré performed in [17]. This leads us to the following question: can Joordens and Paré's peer review method be integrated into computer science courses as a way to teach code review?

We propose that Joordens and Paré's experiment be repeated—this time, in a computer science course. While some researchers might balk at the prospect of modifying the peerScholar system to facilitate code review, we propose an alternative solution. Since 2008, The University of Toronto Department of Computer Science has been developing an online marking tool called MarkUs. MarkUs streamlines the process of submission collection, grading, and returning the feedback to the students. MarkUs is currently in use at the University of Toronto, the University of Waterloo, and L'Ecole Centrale de Nantes in France [18]. MarkUs could be modified to enact the peerScholar workflow, and enable students to evaluate one another.

We would then apply Joorden's and Paré's analysis techniques in order to test whether or not their results repeat themselves in the computer science course context. This, coupled with student feedback, could give valuable insight as to whether or not this workflow is pedagogically useful and practical for computer science courses.

7.3 Is the performance boost a learning effect?

As mentioned in Chapter 6, we cannot be certain if the performance boost is an example of priming, a learning effect, or something else entirely. We would attempt to answer this question by having students perform peer code review several times over the course of several weeks, and then have them produce submissions for grading some weeks after the code review sessions are finished. We would then compare the grades this group receives from teaching-assistants with the grades from a control group that does not perform any peer code review. A performance boost in this case would suggest a learning effect, as opposed to a priming effect.

7.4 What is the impact of giving feedback on feedback?

Suppose we have a series of submissions for a subject to mark, one at a time. After the subject completes marking each submission, suppose we return feedback based on how close the marks they gave matched those given by an “expert” marker. Do the subjects tend to get better at marking after receiving the feedback?

We would test this in a similar fashion as described in Chapter 3. Both the treatment and the control group would perform multiple rounds of grading, but the treatment group would receive feedback for their grading, and the control group would not. We would then measure the ability of each group to approximate marks given by the expert marker.

7.5 Does the amount of time spent reviewing impact the learning benefit?

To answer this question, we would carry out an experiment very similar to the one described in Chapter 3, but we would discard the control and treatment groups. Instead, we would have several groups that all perform grading, but each group has a different limit on the amount of time they can spend grading.

We would then test to see if there is a relationship between the amount of time spent grading, and the performance on the second assignment. We would also take the opportunity to see if the peer grade average correlation is affected by the amount of time spent grading.

7.6 What's the minimum number of peer evaluations needed in order to calculate a peer average?

We found that the peer average mark for a submission was not significantly different from the kind of mark that an expert marker (in our case, a graduate-level teaching assistant) would give. We aimed to have 5 peer graders per mock-up to compute the peer average from. The number 5 was chosen arbitrarily (see Section 4.2).

But what is the minimum number of subjects required to compute a peer average with the robustness and reliability of an expert marker?

We would test this question by having a groups of subjects perform peer grading on a mocked up submission, and then determine what the minimum number of peer grades can be randomly selected from the group in order to get a peer average that is as reliable as an expert grader.

7.7 Is there a relationship between the performance on the first assignment, and the performance increase on the second assignment after the peer grading?

Is it the case that subjects with weaker programming abilities benefit more from peer grading than subjects with stronger programming abilities? Or is it the other way around? Or does it not matter?

We would, again, test this using a similar experiment to the one described in Chapter 3. We would discard the control group in favour of all subjects performing peer grading. We would then take the average mark on the first assignment, and determine whether or not the students who were below average received more of a performance boost than the students who were average or above average.

7.8 Do students get better at grading over time?

Over some length of time, we would have subjects come in periodically and perform peer grading on mocked up submissions. We would then check the correlation between the marks they give and the marks from an expert grader, and see if the correlation gets stronger over time.

7.9 Does peer grading make students better at doing harder assignments?

Our experiment detected a performance increase for two assignments of relatively similar difficulty. What happens if the second assignment is more difficult? Do we still see an improvement?

Once again, we would test this using a similar experiment to the one described in Chapter 3. However, we would have all subjects perform peer grading after their first assignment. After the grading is completed, half of the subjects will complete a more difficult assignment, while the other half would complete an assignment of similar difficulty to their first assignment. We would then test to see if the performance increase persists across both difficulties.

Chapter 8

Conclusion

This study has explored the effects of peer code review on the programming abilities of students. We can now offer plausible answers to the research questions first introduced in Chapter 2.

RQ1 : Does the act of *performing* code review cause students to become better programmers? If so, in what ways do they become better?

Our results show that peer code review does cause a general increase in programming performance for students. This increase is small, yet significant.

RQ2 : To what degree do peer graders agree with graduate-level teaching assistants with respect to assigning marks to assignment submissions?

The peer grade average of the total submission mark generated by the peer graders seemed to be in agreement with the two graduate-level graders.

RQ3 : What do computer science students think about peer grading with respect to their programming assignments?

Students tended to believe that seeing their peers' code taught them something new, regardless of their reported experience level. Students also tended to enjoy the act of seeing and reading their peers' code, and believed that seeing their peers' code gave them a better sense of their own code quality. However, students tended not to enjoy the act of grading their peers' code, and were not always confident in their grading abilities.

We hope that this is just one of the first steps in exploring this research area. We believe that the recent interest in peer assessment can and should be spread into the computer science discipline, as a possible way of teaching code review. We have outlined avenues for future research based on what we have found, the results of which will hopefully benefit computer programming students in the years to come.

Bibliography

[1] B.S. Bloom. *The Taxonomy of Educational Objectives, The Classification of Educational Goals*. D. McKay Co, 1956.

[2] Sue Bloxham and Amanda West. Understanding the rules of the game: marking peer assessment as a medium for developing students' conceptions of assessment. *Assessment & Evaluation in Higher Education*, 29:721–733, 2004.

[3] B. W. Boehm and P. N. Papaccio. Understanding and controlling software costs. *IEEE Transations on Software Engineering*, 14(10):1462–1477, 1988.

[4] Patricia Cartney. Exploring the use of peer assessment as a vehicle for closing the gap between feedback given and feedback used. *Assessment & Evaluation in Higher Education*, 35:551–564, 2010.

[5] Dr. J. Clarke. Re: Mike Conley on grader (dis)agreement. private communication.

[6] Jason Cohen. *Best Kept Secrets of Peer Code Review*. Smartbearsoftware.com, 2006.

[7] M. E. Fagan. Design and code inspections to reduce errors in program development. *IBM Systems Journal*, 38(2):258–287, 1999.

[8] Nancy Falchikov. Involving students in assessment. *Psychology Learning and Teaching*, 3, 2003.

[9] Nancy Falchikov. The place of peers in learning and assessment. In David Boud and Nancy Falchikov, editors, *Rethinking Assessment in Higher Education : Learning for the Longer Term*, pages 135–154. Routledge, 2007.

[10] Nancy Falchikov and Judy Goldfinch. Student peer assessment in higher education: a meta-analysis comparing peer and teacher marks. *Review of Educational Research*, 70:287–322, 2000.

[11] Kenneth I. Forster and Chris Davis. Repetition priming and frequency attenuation in lexical access. *Journal of experimental psychology: learning, memory, and cognition*, 10:680–698, 1984.

[12] Stuart A. Fry. Implementation and evaluation of peer marking in higher education. *Assessment and Evaluation in Higher Education*, 15:177–189, 1990.

[13] Edward F. Gehringer. Electronic peer review and peer grading in computer-science courses. In *Proceedings of the 32nd SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '01, pages 139–143, New York, NY, USA, 2001. ACM Press.

[14] Christopher Hundhausen, Anukrati Agrawal, Dana Fairbrother, and Michael Trevisan. Integrating pedagogical code reviews into a cs 1 course: an empirical study. *SIGCSE Bull.*, 41:291–295, March 2009.

[15] Douglas N. Jackson and Samuel Messick. Acquiescence and the factorial interpretation of the MMPI. *Psychological Bulletin*, 58:299–304, 1961.

[16] Jianguo Liu and Dawn Thorndike Pysarchik. Peer review in the classroom. *Bioscience*, 52:824–829, 2002.

[17] Dwayne E. Paré and Steve Joordens. Peering into large lectures: Examining peer and expert mark agreement using peerscholar, an online peer assessment tool. *Journal of Computer Assisted Learning*, 24(6):526–540, 2008.

[18] The MarkUs Project. MarkUs Project. `http://www.markusproject.org`.

[19] Tim S. Roberts. Self, peer and group assessment in e-learning: An introduction. In Tim S. Roberts, editor, *Self, Peer and Group Assessment in E-Learning*, pages 1–16. Idea Group Inc., 2006.

[20] Philip M. Sadler and Eddie Good. The impact of self- and peer-grading on student learning. *Educational assessment*, 11:1–31, 2006.

[21] Keith Topping. Peer assessment between students at colleges and universities. *Review of Educational Research*, 68:249–276, 1998.

[22] Scott A. Turner. *Peer Review in CS2: the Effects on Attitudes, Engagement, and Conceptual Learning.* PhD dissertation, Virginia Polytechnic Institute and State University, 2009.

[23] Scott A. Turner, Manuel A. Pérez-Quiñones, Stephen Edwards, and Joseph Chase. Peer review in cs2: conceptual learning. In *Proceedings of the 41st ACM technical symposium on Computer science education*, SIGCSE '10, pages 331–335, New York, NY, USA, 2010. ACM.

[24] Scott A. Turner and Manuel A. Pérez-Quiñones. Exploring peer review in the computer science classroom. *CoRR*, abs/0907.3456, 2009.

[25] Anne Venables and Raymond Summit. Enhancing scientific essay writing using peer assessment. *Innovations in Education and Teaching International*, 40:281–290, 2003.

[26] Yanqing WANG, LI Yijun, Michael Collins, and Peijie LIU. Process improvement of peer code review and behavior analysis of its participants. In *Proceedings of the 39th*

SIGCSE technical symposium on Computer science education, SIGCSE '08, pages 107–111, New York, NY, USA, 2008. ACM.

[27] Richard L. Weaver. Peer evaluation: A case study. *Innovative Higher Education*, 11:25–38, 1986.

[28] Karl E. Wiegers. *Peer Reviews in Software: A Practical Guide.* Addison-Wesley Professional, 2002.

Chapter 9

Appendix

We now present tables that were moved out of Chapter 5 in order to improve pacing and increase readability.

9.1 Grader and peer grader agreement tables

Tables 9.1 and 9.2 show the correlations between the average mark per mock-up given by both graders (G1 and G2) and the peer average (PA) per assignment. The particularly strong correlations ($p < 0.05, N = 5$) are marked with a **. The weak correlations are marked with a *. Some correlations could not be calculated due to a zero standard deviation. These are indicated by DIV/0!.

9.2 Comparing grader and peer grader agreement

Like Joordens and Paré in [17], we used Fisher's Z Transformation to compare correlations between both graders and the peer average. With Fisher's Z Transformation, significant difference is found between two correlations if when the Z value exceeds 1.96. This only occurs once, with the add_passenger design criterion for Flights and Passengers (marked with a *).

Criterion	PCC G1/G2	PCC G1/PA	PCC G2/PA
Deck constructor correctness	DIV/0!	DIV/0!	DIV/0!
Deck constructor design	0.65	0.69	0.91
__str__ correctness	DIV/0!	DIV/0!	DIV/0!
__str__ design	0.68	0.88**	0.64
deal correctness	0.41	0.01*	-0.52*
deal design	0.26	0.83	0.46
shuffle correctness	DIV/0!	DIV/0!	DIV/0!
shuffle design	0.97**	0.91**	0.77
cut correctness	0.98**	0.90**	0.93**
cut design	0.31	0.80	0.64
Error checking	0.97**	0.95**	0.96**
Style	0.89**	0.74	0.86
Docstrings	0.65	0.52	0.89**
Internal comments	0.56	0.95**	0.63
Total submission mark	0.81	0.86	0.98**

Table 9.1: Grader (G1 and G2) and peer average (PA) agreement on Decks and Cards using Pearson's Correlation Coefficient (PCC)

Criterion	PCC G1/G2	PCC G1/PA	PCC G2/PA
Flight constructor correctness	0.92**	0.57	0.62
Flight constructor design	0.50	0.39	0.82**
__str__ correctness	DIV/0!	DIV/0!	0.16*
__str__ design	0.53	0.54	0.69
add_passenger correctness	DIV/0!	DIV/0!	DIV/0!
add_passenger design	1.00**	0.89**	0.89**
heaviest_passenger correctness	0.96**	0.69	0.76
heaviest_passenger design	0.94**	0.90**	0.84**
lightest_passenger correctness	0.96**	0.36	0.56
lightest_passenger design	0.94**	0.90**	0.84**
Error checking	DIV/0!	DIV/0!	DIV/0!
Style	0.07*	0.16*	0.56
Docstrings	0.93**	0.91**	0.83**
Internal comments	1.00	0.98**	0.98**
Total submission mark	0.92**	0.97**	0.96**

Table 9.2: Grader (G1 and G2) and peer average (PA) agreement on Flights and Passengers using Pearson's Correlation Coefficient (PCC)

Criterion	Z for G1/G2 vs. G1/PA	Z for G1/G2 vs. G2/PA
Deck constructor correctness	DIV/0!	DIV/0!
Deck constructor design	-0.07	-0.75
__str__ correctness	DIV/0!	DIV/0!
__str__ design	-0.55	-0.07
deal correctness	0.43	1.01
deal design	-0.92	-0.23
shuffle correctness	DIV/0!	DIV/0!
shuffle design	0.56	1.07
cut correctness	0.83	0.64
cut design	-0.78	-0.44
Error checking	0.26	0.15
Style	0.47	0.13
Docstrings	0.20	-0.65
Internal comments	-1.20	-0.11
Total submission mark	-0.17	-1.17

Table 9.3: Fisher's Z Transformation, comparing correlations between both graders (G1 and G2) and the correlations between each grader and the peer average (PA).

Some correlations could not be calculated due to a zero standard deviation. These correlations clearly cannot be compared with, and are indicated by DIV/0!.

The Fisher's Z Transformations for both assignments can be found in Table 9.3 and 9.4.

Criterion	Z for G1/G2 vs. G1/PA	Z for G1/G2 vs. G2/PA
Flight constructor correctness	0.94	0.86
Flight constructor design	0.14	-0.61
__str__ correctness	DIV/0!	DIV/0!
__str__ design	-0.01	-0.26
add_passenger correctness	DIV/0!	DIV/0!
add_passenger design	2.38*	2.38*
heaviest_passenger correctness	1.10	0.95
heaviest_passenger design	0.27	0.52
lightest_passenger correctness	1.57	1.31
lightest_passenger design	0.27	0.52
Error checking	DIV/0!	DIV/0!
Style	-0.09	-0.56
Docstrings	0.13	0.47
Internal comments	1.50	1.50
Total submission mark	-0.50	-0.36

Table 9.4: Grader (G1 and G2) and peer average (PA) agreement on Flights and Passengers using Pearson's Correlation Coefficient (PCC)

	Grader 1		Grader 2	
	Control	Treatment	Control	Treatment
Difference in Quality Mean	2.33	2.67	2.6	2.67
Difference in Quality Standard Deviation	1.05	1.05	0.91	0.90
P-Value	0.42		0.86	

Table 9.5: Mark deltas on all criteria

	Grader 1		Grader 2	
	Control	Treatment	Control	Treatment
Second Assignment Preferred	33.33%	66.67%	46.67%	73.33%
P-Value	0.14		0.26	

Table 9.6: Mark deltas on all criteria

9.3 Grader preference tables

We measured grader preference by using Fisher's Exact Test. Table 9.5 shows that the graders did not detect a significant increase in quality from the first assignment to the second when looking at submissions from both the control and treatment groups.

Table 9.6 shows that both graders had a tendency to prefer the second assignment over the first assignment in the treatment group, as opposed to the control group. However, analyzing the proportion of selections with Fisher's Exact Test does not reveal significance at the .05 level. This finding is, therefore, inconclusive.

	1—Strongly Disagree	2	3	4	5—Strongly Agree
Number of Subjects	1	1	4	2	**7**
Average Months Experience	1	5	13	16	16

Table 9.7: Number of months of experience compared with responses to "Seeing my peers' code taught me things I didn't already know."

9.4 Number of months of experience compared with responses to "Seeing my peers' code taught me things I didn't already know."

Table 9.7 shows the average number of months of experience for the students the responded to the statement "Seeing my peers' code taught me things I didn't already know". The conclusion we draw from this is that the subjects seem to believe that peer grading teachers them something—even if that something is just another approach to the same problem. This seems to be true for subjects regardless of their reported experience level.