The Relationship Between Voluntary Practice of Short Programming Exercises and Exam Performance

Stephen H. Edwards, Krishnan P. Murali, and Ayaan M. Kazerouni
Virginia Tech, Department of Computer Science
Blacksburg, VA
edwards@cs.vt.edu,metarus208@gmail.com,ayaan@vt.edu

ABSTRACT

Learning to program can be challenging. Many instructors use drill-and-practice strategies to help students develop basic programming techniques and improve their confidence. Online systems that provide short programming exercises with immediate, automated feedback are seeing more frequent use in this regard. However, the relationship between practicing with short programming exercises and performance on larger programming assignments or exams are unclear. This paper describes an evaluation of short programming questions in the context of a CS1 course where they were used on both homework assignments, for practice and learning, and on exams, for assessing individual performance. The open-source drill-and-practice system used here provides for full feedback during practice exercises. During exams, it allows limiting feedback to compiler errors and to a very small number of example inputs shown in the question, instead of the more complete feedback received during practice. Using data collected from 200 students in a CS1 course, we examine the relationship between voluntary practice on short exercises and subsequent performance on exams, while using an early exam as a control for individual differences including ability level. Results indicate that, after controlling for ability, voluntary practice does contribute to improved performance on exams, but that motivation to improve may also be important.

CCS CONCEPTS

• **Social and professional topics** → **Computer science education**; **CS1**; **Student assessment**; • **Applied computing** → **Interactive learning environments**; *Computer-managed instruction.*

KEYWORDS

programming exercises; homework; coding; skill development; practice; exam

ACM Reference Format:
Stephen H. Edwards, Krishnan P. Murali, and Ayaan M. Kazerouni. 2019. The Relationship Between Voluntary Practice of Short Programming Exercises and Exam Performance. In *ACM Global Computing Education Conference 2019 (CompEd '19), May 17–19, 2019, Chengdu,Sichuan, China.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3300115.3309525

CompEd '19, May 17–19, 2019, Chengdu,Sichuan, China

ACM ISBN 978-1-4503-6259-7/19/05…$15.00
https://doi.org/10.1145/3300115.3309525

1 INTRODUCTION

A number of drill-and-practice tools have been developed to help students build their skills with basic programming techniques while improving their programming confidence. Despite the presence of some experimental research on the impact of small programming exercises [10, 15], many tools have not been evaluated for impact, particularly in the face of the various choices instructors can make about how to employ them in class. This paper explores the question of whether *voluntary practice* on small programming questions affects student performance, as measured by summative exam scores. While it seems obvious that practice would help, one significant issue is that in a voluntary practice situation, the students who choose to practice are completely self-selected. Thus, it is difficult to separate out any potential gains due to practice from other individual traits that may lead them to choose to practice, and that might also lead to improved performance.

This research was conducted using CodeWorkout [6], an online drill-and-practice system designed to provide small-scale practice assignments in the contexts of both individual learning, and learning in the CS classroom. It is a completely online and open system and is not limited to short single-method programming questions but is capable of supporting different kinds of questions, including multiple choice (both forced choice and multiple answer), coding by filling in the blanks, using arbitrary objects (lists, maps, or even instructor-defined classes) instead of only primitives, writing collections of methods or an entire class (instead of just single functions), multi-part questions that include multiple prompts, and "find and fix the bug" style questions where students are given a code implementation containing one or more errors to repair.

In this paper we build on the work presented in [6] by reporting on a study of the use of short programming exercises in a CS1 course, including both required exercises completed for credit, and optional ungraded practice exercises. Using one course exam to estimate student academic ability before practice is available, and a second to assess performance afterward, we find that opting to practice non-graded exercises is associated with a statistically significant increase in performance, independently of student ability. In contrast, scores on the first exam were not a strong predictor of the choice to practice, indicating that this is not simply an issue of "strong students" performing better on multiple tasks, and choosing to practice simply because they are higher performers.

Section 2 discusses related work and Section 3 gives a brief summary of CodeWorkout. Section 4 describes the study's subjects, method, data, and analysis used to explore student practice.

2 RELATED WORK

Coding systems have begun to emerge specifically to support practice at programming. Instead of questions, these systems ask students to solve problem descriptions through code. CodingBat [14] (formerly JavaBat) offers a collection of small programming problems and now also supports instructor-contributed problems. CodingBat takes advantage of test cases to evaluate the correctness of students' code. CodeWrite [5] similarly evaluates student code, but specifically holds students responsible for writing exercises and their respective test cases.

When students practice the exercises in either system, the only feedback they receive is whether each test case passed or failed. Identifying the failed test cases help students revise their solutions. However, by exposing the test cases, the systems no longer require students to think critically about what situations their code needs to consider. Instead, the feedback of failed/passed test cases isolates a path-of-least-resistance to the solution. The purpose of drill-and-practice is to learn problem solving. Therefore, it would be more beneficial to provide students with guidance rather than just making the solution easier to recognize.

Estey et al. developed BitFit [11] to provide a platform for CS 1 students to engage in voluntary programming practice. The practice was voluntary because usage of the system did not contribute to course grades. Using log data from the system, Estey et al. were able to explore the relationship between this voluntary practice and final exam performance [10]. Analysis showed negative correlations between the number of hints requested and final exam performance: low-scoring students requested more hints than mid-scoring students, who requested more hints than high-scoring students. There was no relationship observed between time spent practicing and final exam performance.

Spacco et. al developed the online coding practice tool CloudCoder [13], which was part of the inspiration for CodeWorkout. Using usage data from several universities [15], analysis showed that the number of programming sessions in CloudCoder was associated with higher performance on exercises. The study also found that the number of exercises completed and attempted, and the percentage of exercises completed were weakly correlated with final exam performance. Correlations with exam performance were even weaker when practice was optional (R^2 = 0.060–0.138) compared to required (R^2 = 0.149–0.295).

Neither [10] nor [15] describe the use of controls for individual differences, or any analytical approach to differentiate between "strong" and "weak" students in their analyses. As a result, correlations might be explained by an uncontrolled individual trait (such as academic strength, prior experience, good study habits, etc.). For example, stronger students who perform well on exams might simply be choosing to practice more because they are higher performers (or might opt out of voluntary practice because they are confident in their abilities). That is, students' tendency to practice and their exam performance might *both* be results of some other unaccounted for effect related to their ability. This makes the reported correlations harder to interpret. To account for this in our analysis (Section 4), we use an early exam as a proxy for student ability, and then investigate the effect of voluntary code writing practice on later exam performance.

Related to code writing exercises, researchers have studied the effectiveness of using Parsons problems for novice programming practice [7, 8]. In [8], Ericson et al. compared students who practiced code-writing, code-fixing, and Parsons problems, and found no differences among the groups in terms of learning performance or cognitive load. A subsequent study used a system with adaptive Parsons problems [7] – problems provided implicit hints when asked for, and had their difficulties adjusted based on the student's performance on the previous problem. The study confirmed that Parsons problems (adaptive and non-adaptive) are just as effective as code-writing exercises in terms of learning gains based on a pre- and post-test. Both studies found that Parsons problems (adaptive and non-adaptive) took less time than code-writing exercises. In our study, we consider a mixture of code-writing and multiple-choice questions, and we do not consider the time taken to complete exercises for reasons described in Section 4.

Perhaps the most common and versatile types of practice systems are those that support free response (FR) and multiple-choice questions (MCQ). Since neither question format is domain-specific, these systems have been adapted in a variety of different fields. However, the versatility of this format also introduces limitations. The developers of StudySieve confirmed that FR answers are difficult to evaluate automatically and consequently had to rely on students to provide feedback [12]. MCQs suffer from the opposite problem: answers are constrained to only a few options so instead it is a challenge to write questions that will evaluate non-trivial knowledge [1]. Furthermore, this broad approach does not lend itself well to providing assistance to support learning specific skills. Despite the shortcomings with its MCQ format, PeerWise takes a novel approach to developing content [2]. PeerWise concentrates on the benefits of peer assessment by allowing students to write questions [3]. Additionally, students review each others' questions and write evaluative feedback. However, Denny identifies a need for external motivators for students to contribute content [4]. While writing and evaluating questions can activate higher order thinking skills, the degree to which these activities constructively contribute to the drill-and-practice environment itself is unknown.

3 CODEWORKOUT

CodeWorkout is a completely online and open drill-and-practice system for all those who are interested in teaching programming to their students. For a complete description of its design and features, refer to [6]. CodeWorkout is not limited to short single-method programming questions but is capable of supporting different kinds of questions, including multiple choice (both forced choice and multiple answer) and coding by filling in the blanks. Exercises are available to be either directly used in the public practice area or to be organized into an assignment. The exercise model is polymorphic: an exercise can be of different types like multiple-choice or coding; they can also consist of multiple parts, allowing for a richer variety of questions.

In addition to programming homework assignments, CodeWorkout is designed to fully support classroom use by instructors who wish to use graded assignments in exam-like situations. In these situations, instructors may choose to impose time limits and limit feedback hints from failed test cases. For example, they may limit

feedback to compiler errors, or provide hints about only a few situations under test.

At the same time, it also provides a completely open "free practice" area where anyone, whether enrolled in a course or not (or even signed in or not), can browse and practice a large collection of publicly available exercises–a concept successfully pioneered by CodingBat. CodeWorkout provides full support for both uses. The analysis in this paper focuses on the second use case, i.e., *voluntary* practice by students in a CS 1 course over two semesters at our university.

4 EFFECTS OF PRACTICE

While most educators already acknowledge the value of practice, it is important to examine the impact of such practice on student performance, as well as to capture experiences in using tools that contribute to this impact. Here, we describe out experiences using CodeWorkout in class, together with a study of how it affects student performance as measured by exam scores.

Ericsson [9] summarizes much of the historical research on practice to improve performance and states the critical aspects necessary for practice to be effective:

> The most cited condition concerns the subjects' motivation to attend to the task and exert effort to improve their performance. In addition, the design of the task should take into account the preexisting knowledge of the learners so that the task can be correctly understood after a brief period of instruction. The subjects should receive immediate informative feedback and knowledge of results of their performance. The subjects should repeatedly perform the same or similar tasks.
>
> When these conditions are met, practice improves accuracy and speed of performance on cognitive, perceptual, and motor tasks.

CodeWorkout has been designed to provide immediate feedback to students as they practice, and to allow them to practice on a series of similar tasks. It also provides instructors with the ability to design specific tasks and arrange them into assignments that guide the students' practice activities. This fits directly into Ericsson's definition of *deliberate practice*, where: "the teacher designs practice activities that the individual can engage in between meetings with the teacher," where the activities are chosen by the teacher with the aim of maximizing improvement [9].

Here, our primary question is whether optional (voluntary) practice has measurable impact on student performance, independent of ability level. In the context of deliberate practice, the exercises are still provided by a teacher with the aim of improving performance. However, the "voluntary" choice by the student directly relates to the student's "motivation to attend to the task and exert effort to improve." We hypothesize that students who have this motivation will opt to complete voluntary practice assignments and benefit more, while students who do not opt to participate in voluntary practice will not see the same benefits.

4.1 Population

CodeWorkout has been used in two courses each semester at Virginia Tech during the 2015-2016 academic year, including use by 372 students in a CS1 course during Fall 2015, and 378 students in the same course in Spring 2016. The study reported here focuses on the Spring 2016 semester, where 200 students in CS1 consented to allowing their data to be used for research purposes. During that semester, CodeWorkout was used for graded homework assignments, for optional practice assignments, and for coding questions on in-class proctored and timed examinations. Students also had larger program assignments as well as homework assignments that did not involve programming in this course.

4.2 Method

The study encompassed four separate phases. First, students began working with automatically graded short practice exercises in required homework assignments during a *training* phase. Next, one third of the way through the course an exam was given that was used as a form of *pretest* to control for individual factors differing between students. Then students engaged in a *practice* phase where they participated in both required and optional practice. Finally, two thirds of the way through the course students took a second exam as a *post-test* where effects from practice were demonstrated.

4.2.1 Training. During the first 5 weeks of the course, CodeWorkout was used by students during homework assignments to practice skills solving basic programming problems. Students were required to complete 20 graded exercises. This arrangement ensures that prior to any exams, students already had exposure to CodeWorkout and were familiar with its interface and how to complete questions online. During graded homework assignments, students had unlimited attempts and unlimited time to practice, and were shown the maximum amount of feedback on each exercise–that is, they saw the results of all software tests applied to their answers, and for nearly all software tests, they also saw the full details of test values and expected results. Only a small number of software tests did not expose the details of what was being tested.

In this situation, most students worked on their solutions until they received a perfect score on an exercise. No penalty was associated with this approach to practicing. Average scores for graded homework were extremely high (96–100%), since most students received full marks on every exercise after sufficient effort. This raises a problem, however, in that scores on such an assignment may be poor predictors, since nearly all students received the same final score, regardless of ability. Only students who allowed themselves insufficient time, or who gave up on exercises without seeking coaching or assistance from the course staff, or who opted not to participate in the assignment at all, received less than perfect scores.

4.2.2 Pretest. One third of the way through the academic term, students took a regularly scheduled exam covering the material learned so far. The exam was held in class and limited to 50 minutes. The test consisted of a number of multiple choice or short answer questions given as an online quiz through the course's learning management system (worth 72% of the exam grade), together with a pair of code writing exercises given using CodeWorkout (worth 28%

Table 1: CodeWorkout assignments in CS1

Assignment	Exercises	Students	Avg. attempts per exercise	Avg. score
Training phase				
Required Homework A	10	197	1.4	100%
Required Homework B	10	176	6.7	98.6%
Pretest phase				
Exam 1	2	198	7.7	84.8%
Practice phase				
Required Homework C	5	195	9.9	96.0%
Required Homework D	10	195	9.0	93.3%
Voluntary Assignment	10	155	7.0	64.9%
Post-test phase				
Exam 2	2	190	11.3	63.7%

of the exam grade). Students saw the online test as a single online activity, with direct links to the CodeWorkout exercises embedded among the other questions of the exam.

During the exam, however, students completed code writing questions under different constraints than during homework. Students had hard time limits and were expected to complete their code writing exercises along with all of the non-coding questions that were also on the exam. In addition, CodeWorkout did *not* give full feedback to students during the exam. Instead, exercises showed compilation errors and limited test results to only three provided examples that were part of the question prompt, keeping all other testing results hidden. Students were expected to judge for themselves whether their answers behaved as intended. Although exercises included an extensive set of tests to assess the correctness of student answers, they could not see the results for these tests or the numeric scores for individual exercises during the exam. Since student work is automatically saved on CodeWorkout each time they check their work, students were free to work on other parts of the exam and come back to review their work, complete with the most recent results on the limited set of examples, whenever necessary until the exam ended.

One critical question of concern is whether optional practice has measurable impact on student performance, independent of ability level. One would expect that practice does have benefits, but one would also expect that more capable students who are already operating at a high level of skill may also be more likely to opt to practice. To address this issue, we used Exam 1 scores as a proxy measure for student ability. It served as a form of pretest to capture individual factors that affect performance on an exam, rather than as a pretest measuring specific knowledge content. Since Exam 1 covered different content (from the first one third of the course) than Exam 2 (the post-test, which covered content from the second third), we could not directly use differences between the two exam scores as a measure of learning gains. However, by using Exam 1 as a proxy for ability (or other individual differences that significantly affect exam performance), we could employ it as an independent variable in testing hypotheses about impacts on Exam 2 scores.

4.2.3 Practice. During the middle third of the course, students completed two more required assignments consisting of short programming questions on CodeWorkout. These covered the basic knowledge content tested on the second exam that serves as the post-test. The two required assignments contained 15 problems.

In addition, prior to the second exam, students were given a purely optional, ungraded practice assignment (the *Voluntary Assignment*) on CodeWorkout consisting of 10 practice problems. This optional practice assignment is the primary focus of this study. The *Voluntary Assignment* occurred after Exam 1, but prior to Exam 2. Because it was optional, only some of the students elected to attempt it—81.6% of students taking Exam 2 opted to attempt at least one exercise on the practice assignment, while just 35.3% percent attempted every exercise in the practice assignment at least once.

Another important issue is how best to characterize participation in the optional *Voluntary Assignment* that occurred between the two tests. Since students were able to continue working on exercises until they mastered them, absolute scores on exercises have questionable value as predictors of outcomes. While other researchers have used time needed to complete an exercise, that measure is also suspect. While some students may successfully complete an exercise in a small amount of time, what does it mean when a different student takes a longer amount of time to achieve the same result? Are longer times indicative of lower skill, if both students achieve full marks on the same exercise? Or are longer times indicative of more time on task and more effort practicing? A more extensive discussion of time effects appears in [15].

Here, because we are interested in the effects of voluntary practice, we divided the subjects into three groups: the *No-Practice* group (18.4% of students) included all students who did not attempt any exercises on the *Voluntary Assignment* at all; the *Some-Practice* group (46.3%) attempted some but not all *Voluntary Assignment* exercises; and the *Full-Practice* group (35.3% of students) attempted every exercise in the *Voluntary Assignment* at least once. This partitioning is based on Ericsson's [9] observation of the importance of the student's motivation to "exert effort to improve" through practice. The student's actions regarding how much of the optional practice assignment to complete is the direct measure that is most closely associated with their motivation to invest in practice.

4.2.4 Post-test. Finally, students took Exam 2 two-thirds of the way through the semester, following the same structure as Exam 1 with both multiple-choice and code-writing questions. Measures of both the multiple-choice question performance and the code-writing

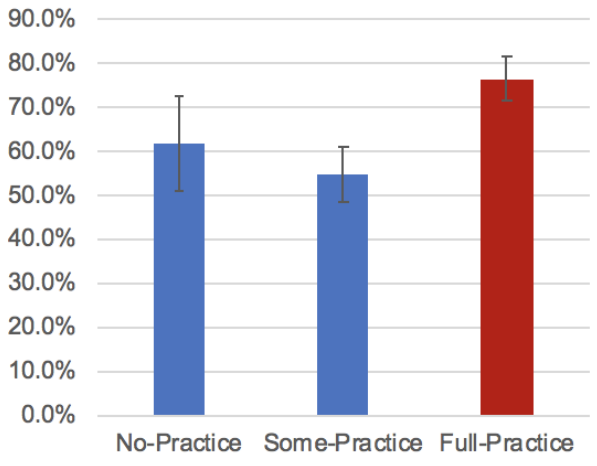

Figure 1: Code writing scores on Exam 2 by group (Full-Practice is significantly different from other groups).

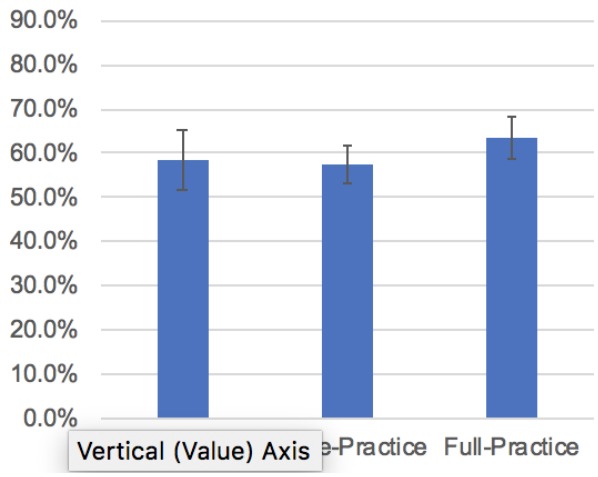

Figure 2: Multiple choice scores on Exam 2 by group (no significant difference).

question performance were analyzed independently to determine if optional practice effects were present. Table 1 shows that Exam 2 scores were lower than Exam 1, which is typical for the course and is due to the larger amount of content knowledge and skills expected two-thirds of the way through the course.

5 RESULTS

Table 1 summarizes CodeWorkout's usage across the four phases of this study. The training phase includes required assignments that ensure a basic level of familiarity with the tool and the style of questions; the first exam serves as a baseline for assessing individual ability; the practice phase includes the *Voluntary Assignment* that is the focus of the study; and the second exam serves as the observation of effects of practice.

5.1 Effects on Exam Code Writing Questions

Since both exams included code writing questions, which presumably require skills similar to those appearing in the practice exercises, as well as other styles of questions that may test other knowledge or skills covered in the course, we can consider the two types of questions separately. On code writing questions answered on CodeWorkout, *Full-Practice* students earned a mean score of 76.5%, compared to 61.5% for *No-Practice* students (s.d. = 31.8%) and 54.7% for *Some-Practice* students (s.d. = 29.7%). Figure 1 illustrates the differences and 95% confidence intervals.

By considering both Exam 1 scores and practice group as independent variables, as well as the cross interaction between them, an analysis of variance indicates a significant effect on the code writing scores in Exam 2 ($df = 189, F = 20.0, p < 0.0001$). Separate effect tests for the two variables indicate both Exam 1 scores ($F = 21.3$, $p < 0.0001$) and practice group ($F = 8.7$, $p = 0.0002$) are significant, but there is no significant interaction between them ($F = 0.4$, $p = 0.69$). The differences in least-squares means indicate that *Full-Practice* is significantly different than the other two groups ($t = 1.97$, $p = 0.049$), but *No-Practice* and *Some-Practice* were not significantly different.

From this analysis, it seems that students who elected to practice some, but not all, of the practice exercises did not perform significantly differently than those who opted not to practice at all. If anything, their scores were lower on all measures (but not significantly). Instead, only students who at least *attempted* all of the practice problems saw significant performance improvements. Most importantly, this improvement is associated with choosing to practice voluntarily, independent of student ability (as measured on Test 1).

To determine effect sizes on code writing scores, partial η^2 values were computed for both the Exam 1 effect and the practice group effect. The effect size for Exam 1 scores is $\eta^2 = 0.264$, indicating that approximately 26.4% of the variance in Exam 2 code writing performance is accounted for by the student's earlier Exam 1 score, which is considered a large effect size. This suggests that some individual traits (such as student ability, prior experience, enjoyment of coding, etc.) may play a role in increasing exam scores independently of voluntary practice. The effect size for the practice group is $\eta^2 = 0.143$, indicating that approximately 14.3% of the variance in Exam 2 code writing performance is associated with whether or not the student chooses to practice all practice problems on the *Voluntary Assignment*, which is also considered a large effect size. As can be seen from the means, Cohen's d for the *Full-Practice* group on code writing exercises in Exam 2 is 0.69.

5.2 Effects on Exam Multiple Choice Questions

On multiple choice and short answer questions, *Full-Practice* students earned 63.6%, compared to *No-Practice* students at 58.6% and *Some-Practice* students at 57.6% (s.d. = 20.5%).

We can also consider the effect on the multiple-choice portion of Exam 2, which does not involve code writing skills directly. Again by considering both Exam 1 scores and practice group as independent variables, as well as the cross interaction between them, an analysis of variance indicates a significant effect on the multiple-choice scores in Exam 2 ($df = 190, F = 14.3, p < 0.0001$). Separate effect tests for the two variables indicate that Exam 1

score ($F = 18.2$, $p < 0.0001$) is significant while practice group ($F = 0.12$, $p = 0.89$) is not. There is no significant interaction between them ($F = 0.10$, $p = 0.90$).

To determine effect sizes on multiple-choice scores, partial η^2 values were computed for the Exam 1 effect. The effect size for Exam 1 scores is $\eta^2 = 0.265$, which is considered a large effect size. Cohen's d for *Full-Practice* on the non-code-writing portion of Exam 2 is 0.28. In addition to being much smaller than the difference on code writing exercises, there is no significant evidence of differences on code writing questions between the practice groups on the multiple-choice and short answer portion of Exam 2.

5.3 Threats to Validity

One major consideration in this study is the potential threat due to individual traits that may affect the choice to opt for voluntary practice. Such individual traits may introduce a form of selection bias that can affect the inferences drawn from correlations between voluntary practice and exam performance. In this study, we used Exam 1 as an indirect measure of individual differences that affect exam performance that could be included in the analysis to see whether the choice to perform additional practice shows an independent effect on exam performance. Greater explanatory power would come from explicitly measuring the most likely individual differences so they could be independently analyzed, but that is beyond the scope of this paper. Still, controlling for individual differences is an important step in establishing whether voluntary practice itself has an independent effect.

At the same time, while the gains experienced by *Full-Practice* students are associated with practice, this study does not provide evidence for the cause of the improvement. Instead, it is clear that the practice alone is insufficient, since students who practiced some, but not all, exercises did not perform significantly differently on any measure from students who participated in no voluntary practice at all. Instead, it appears that the motivation that students have to engage in deliberate practice is also critical, as noted by other researchers and summarized by Ericsson.

Further, this study includes both required and voluntary practice. As shown using CloudCoder [15], requiring short programming exercises is more strongly associated with improvements in exam performance than voluntary practice. While lack of controls on individual differences in that study requires caution, its results suggest that examining the effects of required practice vs. optional practice is warranted. In addition, the CloudCoder study suggests that larger numbers of practice exercises offered in more assignments may offer more impact. The results shown here are likely to be different if more (or fewer) required assignments using more (or fewer) exercises were included, or if a larger number of voluntary exercises were offered, perhaps at more points throughout the course. Additional research is needed to understand the effects of required vs. optional assignments and choices about number of practice opportunities designed by the instructor.

Finally, throughout the short programming exercise community, quality of exercises is a known issue that constantly requires attention. This study tacitly assumes that the exercises employed on practice assignments and on exams are effective. However, given the 50 or so exercises used in this study, some degree of variation is quality is unavoidable. According to the theory of deliberate practice, exercises need to be carefully designed by an expert teacher or coach, and ideally are tailored to the ability level of the student. Potentially, different gains could be achieved with a different set of practice exercises, which is not taken into account in this study.

6 CONCLUSION

CodeWorkout is a new drill-and-practice system designed to provide a larger range of opportunities for students to practice basic coding skills. Inspired by predecessors, it design aims to combine the best strengths of prior work with new strategies for enhancing classroom support and supporting student practice. In the experiences reported in this paper, CodeWorkout was used quite successfully in an introductory course with a large number of students. Students reacted positively and appear to see clear benefits to using this style of exercise practice to develop their skills.

We also conducted an evaluation of how voluntary practice affects student performance, as measured by exam scores. Nevertheless, there are still many research questions needing further exploration. In the future, we plan to continue refining CodeWorkout to support such investigations. An important part of this effort is the use of item response theory to characterize the performance of questions and the identification of questions that may need revision or editorial attention. At the same time, IRT offers a deeper, more data-driven way of estimating student ability. By modeling both student ability and question performance, it is possible to intelligently recommend new problems for practice that are closer to the student's zone of proximal development. The addition of social features and Stack Overflow-inspired Q&A discussions for questions offers a unique strategy to try to help students who get stuck. Similarly, prompting successful students to offer hints on how to tackle questions they have completed, together with analysis of when those hints help later students get unstuck, offer new strategies for engaging students in the community of users, instead of encouraging them to pursue individual practice in isolation. A consolidated approach to these issues is more likely to meet the needs of current and future educators.

Still, the current evaluation does provide some evidence for how voluntary practice affects student exam performance. By using an exam earlier in the course as a proxy for student ability level, we compared how students who practiced all practice problems in an optional practice assignment performed compared to students who did not, while accounting for ability level. While ability level explained a larger amount of variance in Exam 2 scores, improved performance on Exam 2 code writing questions was significantly associated with optional practice, independently of student ability. Further, it is clear that practicing *all* of the optional exercises instead of just some was also important—a behavior we hypothesize as being associated with a student's intrinsic motivation to exert effort to improve, which is a critical component for deliberate practice to be effective. This gives some evidence that the basic intuition of educators—that practice helps, when students engage in it—is associated with better performance, at least with the skills that were practiced, and that this affect may be distinguishable from that of performance gains driven by simple ideas of student ability or pre-existing skill level.

ACKNOWLEDGMENTS

This work is supported in part by the National Science Foundation under grant DRL-1740765. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

[1] Albert Corbett, John Anderson, Art Graesser, Ken Koedinger, and Kurt VanLehn. 1999. Third Generation Computer Tutors: Learn from or Ignore Human Tutors?. In *CHI '99 Extended Abstracts on Human Factors in Computing Systems (CHI EA '99)*. ACM, New York, NY, USA, 85–86. https://doi.org/10.1145/632716.632769

[2] Paul Denny, John Hamer, Andrew Luxton-Reilly, and Helen Purchase. 2008. PeerWise: Students Sharing Their Multiple Choice Questions. In *Proceedings of the Fourth International Workshop on Computing Education Research (ICER '08)*. ACM, New York, NY, USA, 51–58. https://doi.org/10.1145/1404520.1404526

[3] Paul Denny, Andrew Luxton-Reilly, and John Hamer. 2008. The PeerWise System of Student Contributed Assessment Questions. In *Proceedings of the Tenth Conference on Australasian Computing Education - Volume 78 (ACE '08)*. Australian Computer Society, Inc., Darlinghurst, Australia, Australia, 69–74. http://dl.acm.org/citation.cfm?id=1379249.1379255

[4] Paul Denny, Andrew Luxton-Reilly, and John Hamer. 2008. Student Use of the PeerWise System. In *Proceedings of the 13th Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE '08)*. ACM, New York, NY, USA, 73–77. https://doi.org/10.1145/1384271.1384293

[5] Paul Denny, Andrew Luxton-Reilly, Ewan Tempero, and Jacob Hendrickx. 2011. CodeWrite: Supporting Student-driven Practice of Java. In *Proceedings of the 42Nd ACM Technical Symposium on Computer Science Education (SIGCSE '11)*. ACM, New York, NY, USA, 471–476. https://doi.org/10.1145/1953163.1953299

[6] Stephen H. Edwards and Krishnan Panamalai Murali. 2017. CodeWorkout: Short Programming Exercises with Built-in Data Collection. In *Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE '17)*. ACM, New York, NY, USA, 188–193. https://doi.org/10.1145/3059009.3059055

[7] Barbara J. Ericson, James D. Foley, and Jochen Rick. 2018. Evaluating the Efficiency and Effectiveness of Adaptive Parsons Problems. In *Proceedings of the 2018 ACM Conference on International Computing Education Research (ICER '18)*. ACM, New York, NY, USA, 60–68. https://doi.org/10.1145/3230977.3231000

[8] Barbara J. Ericson, Lauren E. Margulieux, and Jochen Rick. 2017. Solving Parsons Problems Versus Fixing and Writing Code. In *Proceedings of the 17th Koli Calling International Conference on Computing Education Research (Koli Calling '17)*. ACM, New York, NY, USA, 20–29. https://doi.org/10.1145/3141880.3141895

[9] K. Anders Ericsson, Ralf T. Krampe, and Clemens Tesch-RÃűmer. 1993. The Role of Deliberate Practice in the Acquisition of Expert Performance. *Psychological Review* 100, 3 (July 1993), 363–406.

[10] Anthony Estey and Yvonne Coady. 2017. Study Habits, Exam Performance, and Confidence: How Do Workflow Practices and Self-Efficacy Ratings Align?. In *Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE '17)*. ACM, New York, NY, USA, 158–163. https://doi.org/10.1145/3059009.3059056

[11] Anthony Estey, Anna Russo Kennedy, and Yvonne Coady. 2016. BitFit: If You Build It, They Will Come!. In *Proceedings of the 21st Western Canadian Conference on Computing Education (WCCCE '16)*. ACM, New York, NY, USA, Article 3, 6 pages. https://doi.org/10.1145/2910925.2910944

[12] Andrew Luxton-Reilly, Paul Denny, Beryl Plimmer, and Daniel Bertinshaw. 2011. Supporting Student-generated Free-response Questions. In *Proceedings of the 16th Annual Joint Conference on Innovation and Technology in Computer Science Education (ITiCSE '11)*. ACM, New York, NY, USA, 153–157. https://doi.org/10.1145/1999747.1999792

[13] Andrei Papancea, Jaime Spacco, and David Hovemeyer. 2013. An Open Platform for Managing Short Programming Exercises. In *Proceedings of the Ninth Annual International ACM Conference on International Computing Education Research (ICER '13)*. ACM, New York, NY, USA, 47–52. https://doi.org/10.1145/2493394.2493401

[14] Nick Parlante. [n. d.]. CodingBat. http://codingbat.com/. last accessed 04-06-2016.

[15] Jaime Spacco, Paul Denny, Brad Richards, David Babcock, David Hovemeyer, James Moscola, and Robert Duvall. 2015. Analyzing Student Work Patterns Using Programming Exercise Data. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education (SIGCSE '15)*. ACM, New York, NY, USA, 18–23. https://doi.org/10.1145/2676723.2677297