An Analysis of Using Many Small Programs in CS1

Joe Michael Allen[1], Frank Vahid[1,2], Alex Edgcomb[1,2], Kelly Downey[1], and Kris Miller[1]
[1]Computer Science and Engineering, University of California, Riverside
[2]zyBooks, Los Gatos, California
jalle010@ucr.edu, vahid@cs.ucr.edu, alex.edgcomb@zybooks.com, kelly@cs.ucr.edu, kmiller@cs.ucr.edu

ABSTRACT

Modern program auto-graders enable new CS1 approaches. Instructors can easily create new assignments, with students receiving immediate score feedback and resubmitting assignments. With such auto-graders, one approach assigns many small programs (MSPs) each week instead of one large program (OLP). Earlier research showed MSPs in CS1 yielded happier students and better grades. Our university and other schools have switched to MSPs in CS1. This paper addresses common questions about MSPs. We analyzed submissions for a 76-student section of our MSP CS1 course. Given 7 MSPs per week each worth 10 points, students needed 50 points for full credit. Students averaged 17 minutes per MSP and 120 minutes per week. Given 7 days, students on average started 2.2 days ahead of the due date, with 37% starting at least 3 days ahead. 40% of students exceeded the required 50 points per week (no extra credit was given). 50% of students "pivoted" -- switching to another program before completing the previous one. 54% used MSPs to study for exams. Students used MSPs in ways beneficial to their learning and stress reduction: spending sufficient time, completing more than necessary, preparing for exams, and pivoting to avoid getting stuck. A common concern is that MSP CS1 students will do poorly in a CS2 using OLPs. We analyzed 5 quarters of CS2 and found MSP students do fine (in fact slightly better). These results encourage use and refinement of MSPs in CS1 and other courses.

CCS CONCEPTS

• Human-centered computing~Empirical studies in HCI • Social and professional topics~CS1 • Social and professional topics~Student assessment • Applied computing~Interactive learning environments • Applied computing~E-learning

KEYWORDS

CS1; MSPs; Auto-grader; Programming; Time spent; Days before due; Threshold; Pivot; Exam preparation; CS2

ACM Reference format:

Joe M. Allen, Frank Vahid, Alex Edgcomb, Kelly Downey, and Kris Miller. 2018. An Analysis of Using Many Small Programs in CS1. *In SIGCSE '19: 50th ACM Technical Symposium on Computer Science Education, Feb 27–Mar 2, 2019, Minneapolis, MN, USA.* ACM, NY, NY, USA, 7 pages. DOI: https://doi.org/10.1145/3287324.3287466

1 Introduction

Student success in CS1 classes is critical to keeping students in the computer science (CS) major, training students in other majors who need some programming, and attracting students to CS. High-stress, poor performance, and negative evaluations in college-level introductory programming classes (CS1) are well known [2, 4, 6]. As such, improving CS1 teaching attracts much research attention, such as peer instruction [7, 8, 9, 10, 11, 14], media focus [3, 7, 10], student self-selection of projects [12], and pair programming [5, 7, 10, 13].

One improvement approach makes use of modern program auto-graders like zyBooks [18], Mimir [16], CodeLab [17], or Cody Coursework [15], to give students immediate feedback, thus allowing for resubmission and improved grades (while conserving limited instructor grading time). Modern commercial auto-graders make assignment creation easier than in the past, causing a dramatic increase in their use in CS1 and other courses; for example, since zyBooks' auto-grader was released in 2016, over 200 courses (mostly CS1) have started using an auto-grader that did not before. With the ease of creating and grading programming assignments, more instructors are creating and assigning many small programs (MSPs) per week rather than the more common one large program (OLP) per week. Our 2018 paper [1] summarized a study showing that MSPs led to happier less-stressed students, without hurting student performance -- and in fact leading to improved code-writing scores on exams, likely due to students having more practice on focused concepts.

This paper's purpose is to answer various common questions about MSPs. This research presents data and analysis on our experience using MSPs in CS1 at our university.

Section 2 describes our methodology, describing our CS1 course and detailing our data collection techniques. Section 3 addresses the question "How much time do students spend working on MSPs?" Section 4 addresses the question "How many days before the due date do students start MSPs?" Section 5 addresses the question "What percent of MSPs do students complete each day?" Section 6 addresses the question "Will students complete more MSPs than required?" Section 7 addresses the question "Do students take advantage of switching among MSPs when stuck (pivot)?" Section 8 addresses the question "Do students use MSPs to study for exams?" Section 9 addresses the question "Do students who learn using MSPs in CS1 do poorly in a CS2 using OLPs?" Section 10 concludes.

2 Methodology

2.1 Course

The study was conducted at our U.S. public research university, whose CS department typically ranks in the top 60 by U.S. News and World Report. The university operates on the quarter system. Each academic year is divided into three "regular" 10-week quarters (fall, winter, spring) and one compressed 5-week summer session. Throughout the academic year, the CS1 course serves around 300-500 students each quarter. The course is required for all computing majors and for various engineering, science, and math majors, such that about half the students are computing majors and half are non-computing majors. The course topics include basic input/output, assignments, branches, loops, functions, and vectors. The weekly structure of the course includes three hours of instructor-led lecture, two hours of TA-led labs, interactive online readings, and auto-graded homework assignments. The course teaches C++ as the programming language. The course has a midterm during week six and a final after week 10. Each exam's points come half from multiple choice questions and half from free-response coding questions. The course uses active learning and peer learning in lectures.

2.2 Data collection

We analyzed data from a Spring 2017 76-student section of our CS1 course that used MSPs. Our CS1 used an online textbook published by zyBooks for all class readings, activities, and programming assignments. At the quarter's end, we collected all student submissions and explores for programming assignments from zyBooks and combined them into one spreadsheet. A submission is defined as when the student "turns in" their assignment for grading. An explore is defined as when a student runs their code through the zyBooks compiler for testing without grading (development was done in the built-in zyBooks coding windows; students were not introduced to an external development environment). Each student submission has metadata about the assignment title, a userID (anonymized and generated from zyBooks), the submission score, the max score possible for the submission, and a timestamp. An explore has the same metadata as a submission but without a score and a max score. For this study, we collected data from the 76 students for 61 MSPs. In total, we collected 16,106 submissions and 48,186 explores for a total of 64,292.

3 How much time do students spend working on MSPs?

We generally expect students to spend about 3 hours per week working on their programming assignments. Our past surveys and analyses showed students on average spending about 2 hours, the average pulled down by students who submit few or no programs (of course some students spend more than 3 hours as well). We designed the MSPs to take about the same total time per week as the traditional OLP approach. A key question is how much time do students actually spend working on MSPs.

To calculate the total time students spent on MSPs, we used each timestamp for an explore or submit, calculated the difference between each timestamp, and summed the differences. We excluded a difference that exceeded 10 minutes, assuming the student took a break. Note that our calculations are thus an underestimate, as some breaks may have actually involved the student working or researching, and we also cannot capture time spent understanding and working on the program before the first explore or submit.

Figure 1 summarizes the average time spent by students on MSPs per week, as calculated above. The x-axis is the week number and the y-axis is the time spent in minutes. On average, students spent 17 minutes per MSP, and 120 minutes per week, excluding week 1 (which had easy introductory programs) and week 9 (which had fewer programs to complete). The two most challenging weeks were week 4 covering loops, and week 8 covering vectors. The dips in weeks 6 and 7 are due to several MSPs having students rewrite earlier MSPs, but using user-defined functions.

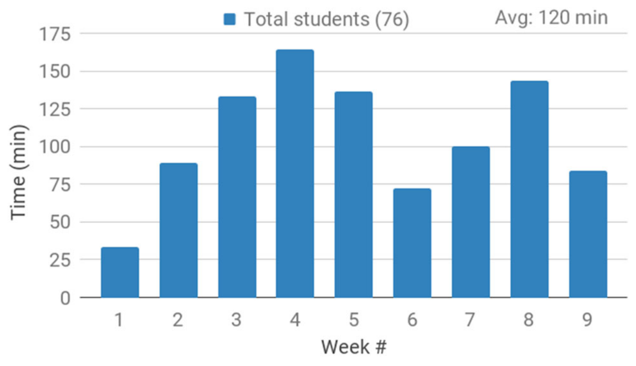

Figure 1: Average time spent by students each week on MSPs. Students with 0 submissions or 0 time spent were excluded from calculations.

We compared our analyses with a survey during lecture of week 8 that had 21 questions, one of which being "*The average hours per week spent on all zyLab programming assignments that week was?*" with response options 1-2, 2-3, 3-4, ..., 10+. Figure 2 summarizes student responses. 67 students responded. A weighted average yields about 5 hours per week, which is higher than our calculated time of 2 hours a week. This higher value may be due to various factors including: our calculations being an underestimate as mentioned earlier, students may overestimate or overreport time spent, weaker students may skip lecture and not be included in the survey, the survey's options may bias students towards selecting higher values, and the weighted sum may unintentionally round up.

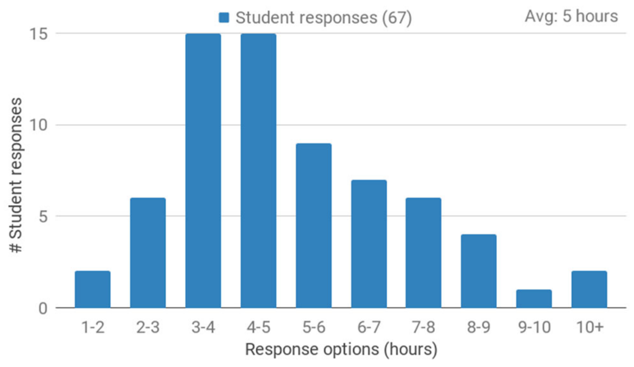

Figure 2: CS1 Spring 2017 survey responses (67 students) for "The average hours per week spent on all zyLab programming assignments that week was?" A weighted sum yields an average of 5 hours per week.

Figure 3 shows the time spent per MSP, using a box-and-whisker plot. The x-axis is the MSP (61 total) and the y-axis is the time spent in minutes. Dashed lines separate MSPs by week. The y-axis is capped at one hour (60 minutes). Students who did not attempt the given MSP are excluded from the calculations.

4 How many days before the due date do students start MSPs?

We released each week's MSPs on Tuesday, all due the following Tuesday at 9:00 pm. That week's readings and lectures (Tuesday and Thursday, 80 minutes each) taught the concepts covered by that week's MSPs. That week's 2-hour lab (Thursday) also taught those concepts, with about 30 minutes at the end for students to work on the MSPs and ask questions. A key question is how many days before the due date do students start working on MSPs.

Figure 4 summarizes the average number of days students began working on MSPs before the due date. The average was computed by finding students' first submission for all MSPs, computing the days between the first submission and the MSP due date, calculating the percent of students that started T-7, T-6, ..., T-0 days before the due date, and then averaging across all MSPs. The x-axis is the number of days prior to the due date. Using "NASA countdown-like" terminology, we use "T-2" to mean two days before the due date (or Sunday). The y-axis is the average percent of students that fall under each category. Week 1 is excluded from these calculations since week 1 MSPs were very easy.

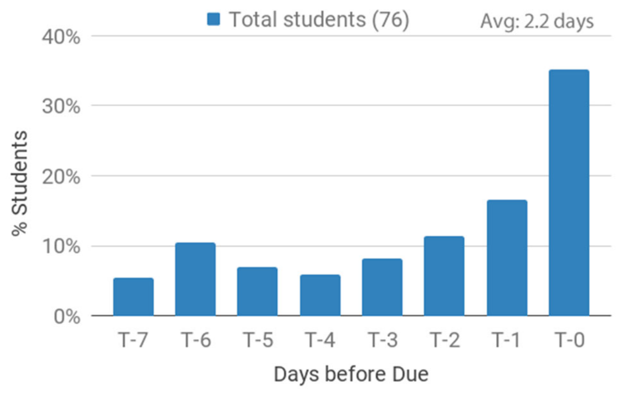

Figure 4: Percent of students who began MSPs each week T-X days prior to the due date - Spring 2017.

To our pleasure, 37% of students (28) started 3 days ahead or more. To our displeasure, 63% of students started only 2 days ahead or less, with 35% of students (27) starting on the due date. Students on average began 2.2 days ahead of the due date.

Figure 5 shows start times for the other two CS1 sections that quarter, which used OLPs. Those students began on average 2.1 days ahead of the due date. Only 28% (48) started 3 days ahead or more, and 25% (43) started on the due date. Note that the due dates were different between the sections, but this comparison still gives valuable insight.

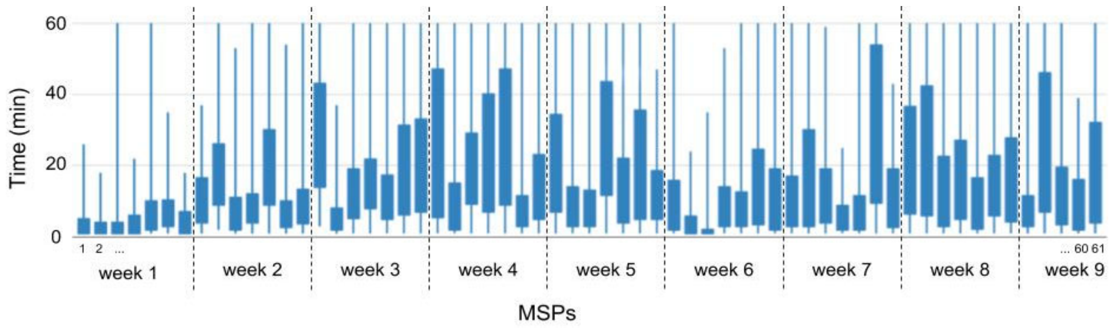

Figure 3: Box-and-whisker plot of student time spent for each MSP. On average, students spent 17 minutes per MSP excluding weeks 1 and 9.

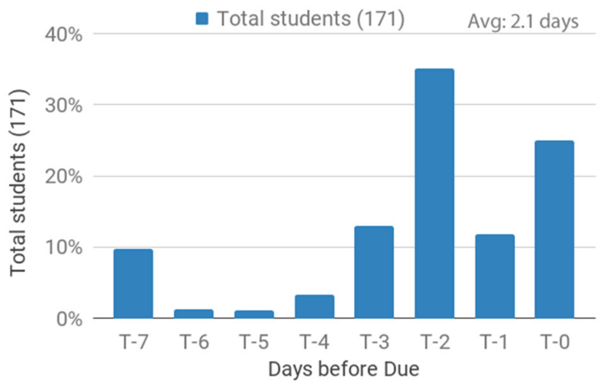

Figure 5: Percent of students who began OLPs each week T-X days prior to the due date - Spring 2017.

We had hoped that MSPs' less-intimidating nature would have led to earlier starts by most students. MSPs had a mild impact on students starting earlier, but many students still started on or near the due date. We believe starting earlier is good practice, and thus decided to try to encourage earlier starts. MSPs made such encouragement easy. In our Fall 2018 course, we simply included the following policy in our syllabus: "To discourage procrastination, you will be required to complete at least 20 points out of the 50 points each week by Sunday at 10 pm", which is 2 days prior to the Tuesday, 10 pm deadline. That small change led to substantial modification in student behavior, with start dates shifting from 2.5 (weeks 2 – 5) to 5.3 days before the due date. At the time of this publication, that Fall 2018 course covered up to week 5, and also excludes week 1 like the earlier data. Future work is needed to see if this improvement also helps to reduce student stress and improve grade performance.

5 What percent of MSPs do students complete each day?

The previous section showed when students started, defined as achieving at least 1 point on the MSP (out of 10 points). Here, we analyze total completion percent per day. A key question is what percentage of MSPs do students complete each day.

Figure 6 summarizes the completion rate of MSPs per day. The x-axis is the number of days prior to the due date and the y-axis is the completion percentage. The top bar is the percent completed on that day and the bottom bar is the cumulative completion prior to that day. Recall that only 50 of 70 points (71%) were required for full credit.

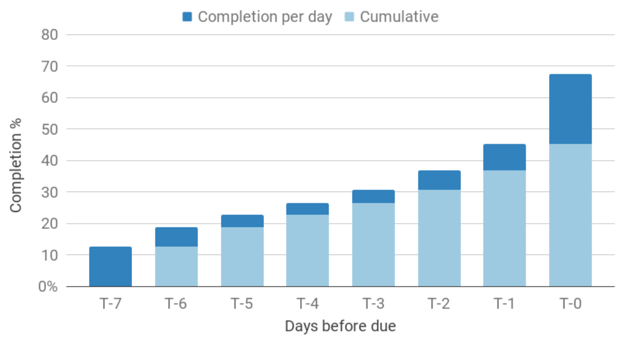

Figure 6: MSP completion T-X days prior to the due date. The top bar is the percent completed on that day, and the bottom bar is the percent completed prior to that day.

Figure 6 shows a gradual increase in the completion rate throughout the week. The completion rate increases 5-10% each day except for the last day (T-0) which has about a 20% increase. Because students need only complete 50 of 70 points, some MSPs have 0% completion, pulling down the averages shown.

6 Will students complete more MSPs than required?

Each week, students were assigned 7 MSPs (10 points each) and were only required to complete 50 points of 70 to score 100% on programming assignments for the week. No extra credit was given for exceeding 50 points. We refer to the 50-point cutoff as the full-credit threshold. A key question is whether students would willingly complete more MSPs than required, which would suggest that they find MSPs useful and/or enjoyable.

Figure 7 shows the percent of students that scored equal to or above the full-credit threshold each week. The bottom bar is the students that completed above the threshold and the top bar is the students that completed equal to the threshold. In weeks 1, 2, 3, and 6, a higher percentage of students scored above the threshold than equal to the threshold. Across the quarter, an average of 40% of students scored above the threshold.

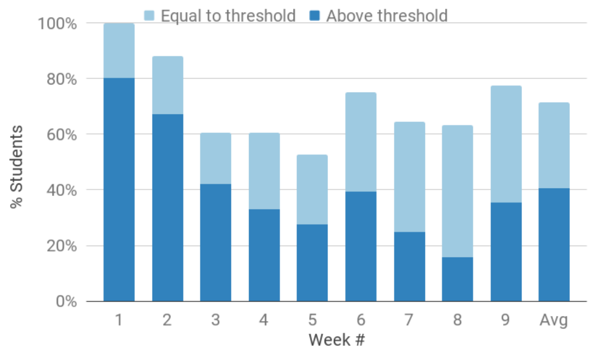

Figure 7: Percent of students who completed equal to or above the full-credit threshold each week.

Figure 8 provides a more detailed analysis via a bubble chart. The x-axis is the week number and the y-axis is the total points scored per week. The bubble size represents the number of students that scored that number of points. For example, the largest bubble in week 1 is labeled 53 because 53 students scored 70 points on MSPs for that week. Note that students who scored 0 points for the week are not included because those students likely dropped the class or decided not to submit labs for the week. The dashed line represents the full-credit threshold for each week. Note that week 9's threshold is lower since only five MSPs were given to students. On average, students who scored more than the full-credit threshold scored an additional 13 points. As each MSP is worth 10 points, this translates to completing an additional 1.3 MSPs each week.

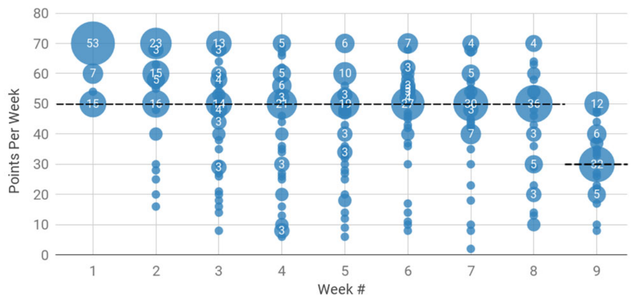

Figure 8: Points students scored each week. Students who scored 0 points for the week are excluded. Dashed line indicates max points for the week.

We were pleased to find that so many students were able to meet the full-credit threshold and that a substantial number were willing to do more than the minimum required work.

7 Do students take advantage of switching among MSPs when stuck (pivot)?

Pivoting is when a student partially completes an MSP (e.g., scores 6 of 10 points) and then decides to work on a different MSP. Typically, with traditional OLPs, students only have the option to work on the program until completion. If stuck, a student has few or no options. With MSPs, the students can pivot to another MSP. A key question is do students take advantage of the opportunity to pivot, and if so how often.

A submission is defined as a pivot if all following rules are met:

1. The current submission is not the student's first submission for the week
2. The current submission is for a different MSP than the previous submission
3. The current submission is for an MSP that has not been completed
4. The previous submission has not been completed
5. The current submission and previous submission are for MSPs assigned in the same week

Figure 9 shows the percent of students who pivoted at least once in a given week. The x-axis is the week number and the y-axis is the percent of students that pivoted that week.

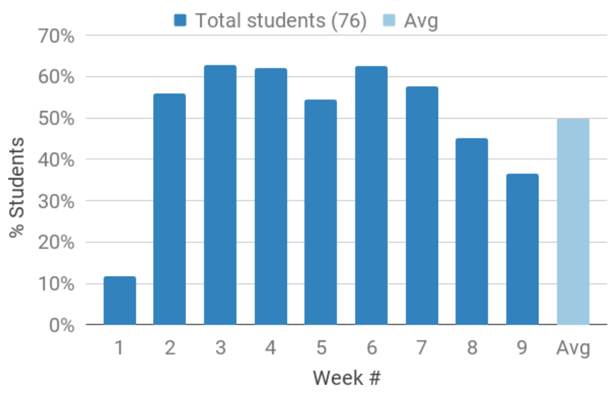

Figure 9: Percentage of students who pivoted at least once in a given week. An average of 50% of students pivoted at least once each week.

We found that students pivot on average 1.3 times each week. The highest number of pivots was one student who pivoted 12 times in week 4. Week 1 had few pivots due to the programs being easy. With more challenging programs beginning in week 2, students made much use of pivots. Students who pivoted at least once a week pivoted on average 2.5 times.

For insight, we highlight three actual pivoting scenarios.

7.1 Pivot at 0% - Week 8 (vectors)

A student attempted MSP 5 three times but received 0 points on all submissions. Instead of continuing MSP 5, the student switched to MSP 7 and scored 10 points. The student did not return to complete MSP 5. The student scored 50 points on MSPs for the week, meeting the 50-point full-credit threshold.

7.2 Single pivot - Week 3 (branches)

A student worked on MSP 4 and scored 8 points. The student switched to MSP 6 and scored 10 points. The student did not return to complete MSP 4. The student scored 48 points on MSPs for the week, nearly meeting the 50-point full-credit threshold.

7.3 Multiple pivots (3 or more) - Week 4 (loops)

A student worked on MSP 4 and scored 2 points. The student switched to MSP 5 and scored 10 points. The student returned to MSP 4 and improved their score from 2 points to 8. The student moved to MSP 7 and scored 9 points. The student then worked on MSP 6 and scored 10 points. Finally, the student returned to MSP 4 and improved their score from 8 points to 10. The student scored 69 points on MSPs for the week, exceeding the 50-point full-credit threshold and nearly hitting the 70-point max.

Students seem to take advantage of the pivot benefit that MSPs offer, especially when a threshold is used. 94% of students (71 students) pivoted at least once throughout the 10-week quarter. As a result, we hope to do future work to investigate whether students who pivot score higher than those who do not, whether there any detriments to pivoting, and whether students who pivot return and solve the MSP they switched away from.

8 Do students use MSPs to study for exams?

Given that MSPs are short, concise, and focus on a single concept, a key question is whether students voluntarily redo MSPs to prepare for exams.

Given the dates for the midterm and final exams, we defined criteria to determine if a student used an MSP for exam practice. We said that a student used an MSP for exam practice if the student had, for that MSP, a submission or explore timestamp that was after the MSP's due date and within one week prior to the exam. The midterm occurred during week six of the quarter and the final occurred at the end of the quarter.

Table 1 shows the results of how many students used MSPs for practice and how many unique MSPs were used to study. 54% of students (41) used MSPs to study for either the midterm or final. 98% of all MSPs (60) were used by at least one student to study for an exam.

Table 1: Student use of MSPs for exam preparation.

Total number of students	76
Total number of MSPs	61
% of students that used MSPs to study for the midterm	38%
% of students that used MSPs to study for the final	37%
% of students that used MSPs to study for either exam	54%
% of MSPs that were used to study for the midterm	97%
% of MSPs that were used to study for the final	90%
% of MSPs that were used to study for either exam	98%

We are pleased to see many students using MSPs to study for exams. For comparison, we looked at the other two sections of CS1 from Spring 2017, which used OLPs. Only 10% of students (17) used OLPs to study for exams.

9 Do students who learn using MSPs in CS1 do poorly in a CS2 using OLPs?

A common concern regarding MSPs in CS1 is the impact MSPs will have on students when they reach CS2 using OLPs. A key question is how do students taught via MSPs in CS1 fare in CS2, compared to students taught via OLPs in CS1.

We gathered data from our CS2 course from Winter 2017 through Spring 2018 (5 quarters). We determined which students took CS1 using MSPs and which took CS1 using OLPs. To be conservative, we excluded students who did not take CS1 at our university. We found 241 students that took MSPs and 312 students that took OLPs. In total, 553 students who took CS2 at our university were considered in our analysis.

Figure 10 shows CS2 performance results. The x-axis shows the class work categories we analyzed (participation activities, labs, programming assignments, midterm exams, final exam, and total grade in the class) and the y-axis is student grade performance. OLP students are the light bars on the left and MSP students are the dark bars on the right.

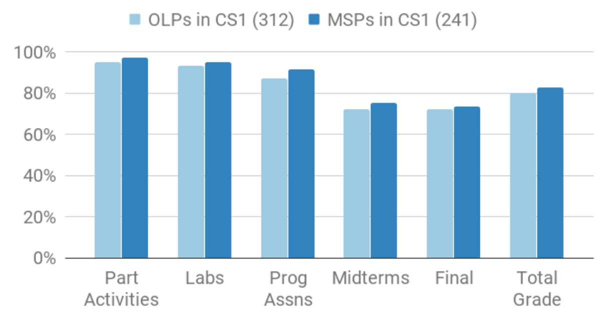

Figure 10: CS2 performance for MSP CS1 students vs. OLP CS1 students. MSP CS1 students do no worse, and in fact do slightly better.

Figure 10 shows that students who took CS1 with MSPs perform similarly, and in fact slightly better, than the students who took CS1 with OLPs. Note that the purpose of this analysis is not to claim MSPs in CS1 lead to better performance in CS2. Instead, the analysis shows that MSPs are not harming students in CS2. We hope to do further research to better understand the effects that using MSPs in CS1 has on students in CS2.

10 Conclusion

Modern easy-to-use auto-graders enable new teaching approaches in CS1 courses, like using MSPs instead of OLPs for weekly programming assignments. Our previous research showed that using MSPs in CS1 yielded happier students and better grades in the course. This paper analyzed how students use MSPs. We conclude that students are making good use of MSPs to aid in their learning process: Students spend sufficient time working on MSPs each week, begin working on MSPs earlier than for OLPs, complete more MSPs than necessary with a full-credit threshold, take advantage of pivoting between MSPs, and use MSPs to study for exams. We also see that MSP CS1 students do just as well, even slightly better, than OLP CS1 students in an OLP CS2. Our department now uses MSPs in all CS1 sections, and we are aware of dozens of other schools that have switched to MSPs as well.

ACKNOWLEDGEMENTS

This work was supported by the U.S. Dept. of Education (GAANN fellowship) and by Google.

REFERENCES

[1] Joe Michael Allen, Frank Vahid, Kelly Downey, and Alex Edgcomb. 2018. Weekly Programs in a CS1 Class: Experiences with Auto-graded Many-small Programs (MSP). In Proceedings of 2018 ASEE Annual Conference & Exposition. DOI: https://peer.asee.org/31231

[2] Theresa Beaubouef and John Mason. 2005. Why the high attrition rate for computer science students: some thoughts and observations. SIGCSE Bull. 37, 2 (June 2005), 103-106. DOI: http://dx.doi.org/10.1145/1083431.1083474

[3] Mark Guzdial. 2003. A media computation course for non-majors. In Proceedings of the 8th annual conference on Innovation and technology in computer science education (ITiCSE '03), David Finkel (Ed.). ACM, New York, NY, USA, 104-108.DOI: http://dx.doi.org/10.1145/961511.961542

[4] Päivi Kinnunen and Lauri Malmi. 2006. Why students drop out CS1 course?. In Proceedings of the second international workshop on Computing education research (ICER '06). ACM, New York, NY, USA, 97-108. DOI: http://dx.doi.org/10.1145/1151588.1151604

[5] Nachiappan Nagappan, Laurie Williams, Miriam Ferzli, Eric Wiebe, Kai Yang, Carol Miller, and Suzanne Balik. 2003. Improving the CS1 experience with pair programming. In Proceedings of the 34th SIGCSE technical symposium on Computer science education (SIGCSE '03). ACM, New York, NY, USA, 359-362. DOI:http://dx.doi.org/10.1145/611892.612006

[6] Andrew Petersen, Michelle Craig, Jennifer Campbell, and Anya Tafliovich. 2016. Revisiting why students drop CS1. In Proceedings of the 16th Koli Calling International Conference on Computing Education Research (Koli Calling '16). ACM, New York, NY, USA, 71-80. DOI: https://doi.org/10.1145/2999541.2999552

[7] Leo Porter and Beth Simon. 2013. Retaining nearly one-third more majors with a trio of instructional best practices in CS1. In Proceeding of the 44th ACM technical symposium on Computer science education (SIGCSE '13). ACM, New York, NY, USA, 165-170. DOI: http://dx.doi.org/10.1145/2445196.2445248

[8] Leo Porter, Cynthia Bailey Lee, and Beth Simon. 2013. Halving fail rates using peer instruction: a study of four computer science courses. In Proceeding of the 44th ACM technical symposium on Computer science education (SIGCSE '13). ACM, New York, NY, USA, 177-182. DOI: http://dx.doi.org/10.1145/2445196.2445250

[9] Leo Porter, Cynthia Bailey Lee, Beth Simon, and Daniel Zingaro. 2011. Peer instruction: do students really learn from peer discussion in computing?. In Proceedings of the seventh international workshop on Computing education research (ICER '11). ACM, New York, NY, USA, 45-52. DOI: http://dx.doi.org/10.1145/2016911.2016923

[10] Leo Porter, Mark Guzdial, Charlie McDowell, and Beth Simon. 2013. Success in introductory programming: what works?. Commun. ACM 56, 8 (August 2013), 34-36. DOI: https://doi.org/10.1145/2492007.2492020

[11] Beth Simon, Michael Kohanfars, Jeff Lee, Karen Tamayo, and Quintin Cutts. 2010. Experience report: peer instruction in introductory computing. In Proceedings of the 41st ACM technical symposium on Computer science education (SIGCSE '10). ACM, New York, NY, USA, 341-345. DOI: http://dx.doi.org/10.1145/1734263.173438

[12] Jeffrey A. Stone and Elinor M. Madigan. 2008. The impact of providing project choices in CS1. SIGCSE Bull. 40, 2 (June 2008), 65-68. DOI: https://doi.org/10.1145/1383602.1383637

[13] Laurie Williams, Kai Yang, Eric Wiebe, Miriam Ferzli, and Carol Miller. 2002. Pair Programming in an Introductory Computer Science. OOPSLA Educator's Symposium, Seattle, WA.

[14] Daniel Zingaro. 2014. Peer instruction contributes to self-efficacy in CS1. In Proceedings of the 45th ACM technical symposium on Computer science education (SIGCSE '14). ACM, New York, NY, USA, 373-378. DOI: http://dx.doi.org/10.1145/2538862.2538878

[15] Cody Coursework. https://coursework.mathworks.com/. Accessed: August, 2018.

[16] Mimir. https://www.mimirhq.com/. Accessed: August, 2018.

[17] Turing's Craft: CodeLab. https://www.turingscraft.com/. Accessed: August, 2018.

[18] zyBooks. https://www.zybooks.com/catalog/zylabs-programming/. Accessed: August, 2018.