An Analysis of Using Coral Many Small Programs in CS1*

Joe Michael Allen, Frank Vahid

Department of Computer Science and Engineering
University of CA, Riverside
jalle010@ucr.edu, vahid@cs.ucr.edu

Abstract

Coral is an ultra-simple programming language designed to look like pseudocode while resembling industry programming languages like C++, Java, and Python. Coral was created specifically for learners and thus, in 2019, our CS1 began teaching programming fundamentals with Coral during the first 3 weeks before switching to C++ for the remainder of the term. Our university already adapted a many small programs (MSP) teaching approach which involves assigning students multiple smaller assignments instead of only giving them one large assignment each week. In this work, we share our experience using a hybrid Coral/C++ MSP approach versus a pure C++ MSP approach. We summarize similarities and differences between student performance and other metrics such as time spent, start date, and more. We found that instructors can use a hybrid Coral/C++ approach to have an easier class startup while maintaining high student grade performance.

1 Introduction

Introductory programming courses known as CS1, are known for their plethora of problems leading to high student stress and high DFW rates. These high numbers are a result of many different factors [1-4]. These issues are especially problematic for CS1 courses as they are essential to keep students in the major, attract new students to programming, and to introduce non-major

students to the basics of programming. Like many other universities around the nation, our university has also struggled to find ways to alleviate these issues. To try and remedy this problem, we began actively pursuing intervention strategies to make our CS1 more accommodating to our students.

1.1 Many small programs (MSPs)

In 2018, we adopted a many small programs (MSPs) teaching approach. An MSP teaching approach involves assigning students multiple smaller programming assignments, typically 5-7, each week instead of only assigning one large programming assignment (OLP) each week. Using an MSP approach allows instructors to give students more assignments to practice programming concepts without overloading them with too much additional work. Previous research [5-6] has shown that using an MSP teaching approach can improve student grade performance and decrease student stress. Other research has shown other benefits such as earlier start dates, good time spent on programming assignments, better exam preparation, and more. We have found success with this approach and have received positive feedback from our students and our CS1 instructors.

1.2 Coral

In 2019, we tried another intervention technique: we taught our CS1 via a hybrid approach of Coral and C++ together. Coral is an introductory web-based, pseudocode-like language designed to help learners [7]. Coral is free to use and resembles popular commercial programming languages like C++, Java, and Python, allowing for a smooth transition between languages. The language comes with a limited set of 7 instructions to help students focus on the fundamentals of programming. Not only is Coral fully executable, it also comes with a flow chart language to help visualize the execution of the code in real-time.

The authors of Coral published an initial work showing Coral's ease of use and we decided to apply the language in our CS1 [8]. We had considered using other introductory programming languages like Snap [9] or Scratch [10], but we found they are not designed for a CS1 class. We began using Coral at the start of the class and then switched midway to C++. This paper is written to share our experiences and findings from our second time using this approach.

2 Methodology

2.1 Course

We analyze a Spring 2020 CS1 course taught at our public research university. Our CS1 typically serves around 300-500 students during a 10-week quarter (fall, winter, spring) split into 3-5 sections of 80 students. All sections use the same zyBooks interactive textbook and require students to complete the same weekly participation activities (class readings), challenge

activities (small coding homeworks), and lab activities (programming assignments). Our CS1 regularly serves half computing major students and half non-major students. The course is taught fully in C++ and covers basic input/output, variables, expressions, branches, loops, functions, and vectors.

2.2 Experiment details

For one CS1 class section (hybrid Coral/C++ group) we taught Coral for the first 3 weeks and then switched to C++ instead of the typical way of teaching C++ for all 10 weeks (pure C++ group). Other differences between each group include the instructors; however, they both have a very similar teaching style and consistently earn similar marks on the end-of-quarter student reviews and the midterm as hybrid group had a few additional Coral related questions. All other class components were the same, including the lesson plan, interactive online textbook, assignment deadlines, etc.

2.3 Data collection

We asked zyBooks to provide us with a detailed log of all student activity for our CS1 class. Student activity consists of develop runs, when a student tests their code using zyBooks' automated system, and submit runs, when a student turns in their code for grading. Each log entry includes the activity name, an anonymized user ID, a score, a max score, and a timestamp.

3 Student grade performance

We gathered gradebooks for each section and to calculate average scores on weekly MSP assignments we gathered all student activity. Students that did not submit any code for grading in a given week were excluded from calculations.

Results: Figure 1 shows our results. The pure C++ group data is shown on the left bars and hybrid Coral/C++ group data is shown on the right bars. The grade percentage is on the y-axis and the week number is on the x-axis. A total grade average column is added to the end of the chart. Table 1 summarizes the average grades for all class categories.

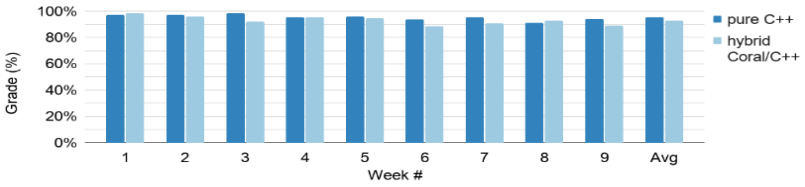

Figure 1: Grade performance results: Both the pure C++ group (avg. 96%) and the hybrid Coral/C++ group (avg. 93%) scored equally well on MSP assignments.

Table 1: Student grade performance on all categories of our CS1 class

Class category	Pure C++	Hybrid Coral/C++
Total class grade	88%	95%
Final exam	83%	88%
Midterm exam	83%	95%
Participation activities	94%	95%
Challenge activities	94%	95%
Lab activities	96%	93%

Figure 1 shows that both the pure C++ group (96%) and the hybrid Coral/C++ group (93%) do equally well on weekly MSP assignments. Table 1 also shows that both groups perform well in all categories of the class, with the hybrid C++/Coral group slightly outperforming the pure C++ group.

4 Weekly MSP assignment metrics

We report results on various metrics related to weekly MSP assignments. For each metric, calculations exclude students that did not attempt any lab activities for the given week. For all charts, the pure C++ group data is shown on the left bars and the hybrid Coral/C++ group data is shown on the right bars.

4.1 Time spent

We expect students to spend around 3 hours working on lab activities each week. To measure student time spent, we summed the differences between each activity timestamp; excluding differences greater than 10 minutes as we considered the student to have taken a break or moved on something else. As such, this data is likely an under-representation as students could have spent that time studying or testing their code outside of the zyBooks IDE.

Results: Figure 2 displays our results. The total time spent is on the y-axis and the week number is on the x-axis. A total time spent average column is added at the end of the chart.

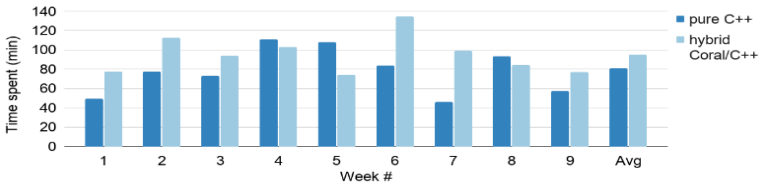

Figure 2: Time spent results: The hybrid group (avg 95min) spends slightly more time working on MSPs each week than the pure C++ group (avg. 81 min).

Figure 2 shows that the pure C++ group (81 minutes) spends less time working on MSPs each week than the hybrid Coral/C++ group (95 minutes).

4.2 Activity runs (develops / submits)

We sought to understand how students develop their code and how frequently students test (develop run) and check (submit run) their code while working. We gathered all student activity and calculated the average number of develop runs and submit runs on weekly MSP assignments.

Results: Figure 3 displays our results. Develop runs are indicated by the dark bars at the bottom and the submit runs by the light bars at the top. The total number of develop/submit runs are on the y-axis and the week number is on the x-axis. A total average column is added at the end of the chart.

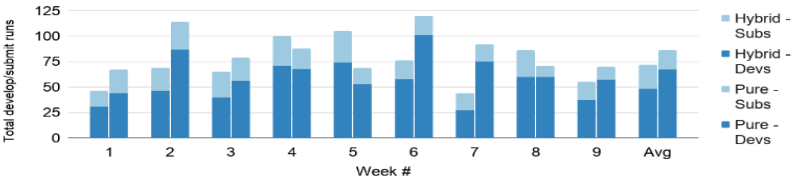

Figure 3: Activity run results: The pure C++ group (avg. 48dev / 24sub) develops less and submits more than the hybrid Coral/C++ group (avg. 67dev / avg. 16 sub).

Figure 3 shows that the pure C++ group develops less than the hybrid Coral/C++ group, but submits more frequently. To fully understand the data, a more in-depth analysis is required; however, since there are more develops than submits on average, it seems like students show a healthy programming practice of testing their code (developing) and then submitting.

4.3 Start date

Each assignment is due one week from the time it is assigned. We consider starting at least 2 days prior to the assignment's due date as healthy behavior. To calculate students' average start date each week, we found each students' earliest activity timestamp, calculated the difference between that and the due date, and averaged the differences.

Results: Figure 4 displays our results. The number of days are on the y-axis and the week number is on the x-axis. A total start date average column and an adjusted total average column is added at the end of the chart to account for a 'grace period' (late submissions allowed) during weeks 1 and 2.

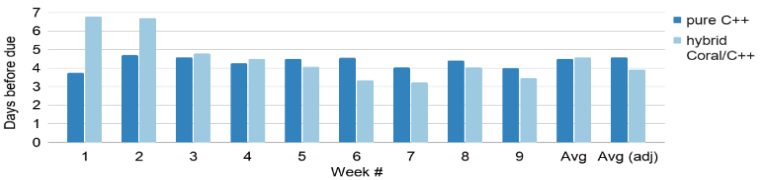

Figure 4: Start date results: The pure C++ group (avg. 4.5days / 4.8days adj.) begins working earlier than the hybrid Coral/C++ group (avg. 4.6days / 3.9days adj.).

Figure 4 shows that both groups begin working about 4.5 days before the due date. Removing weeks 1 and 2 to account for the 'grace period', Figure 4 shows that the pure C++ students begin 4.6 days early whereas the hybrid Coral/C++ students begin 3.9 days early (see 'Avg (adj)' column).

4.4 Pivoting

A pivot is when a student switches from one lab activity to another without completing (scored 100%) the current one they are working on. Pivoting enables students to score additional points when stuck or even use another lab activity to help them solve the current problem they are facing.

Results: Figure 5 displays our results. The total number of pivots are on the y-axis and the week number is on the x-axis. A total pivot average column and total pivot average adjusted column is added at the end of the chart to account for the midterm in week 6.

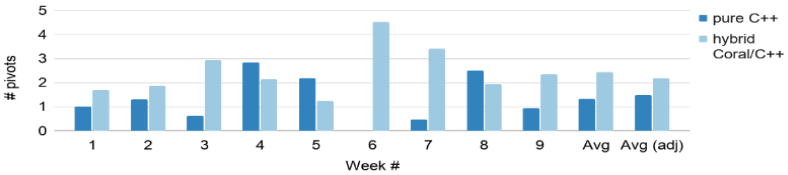

Figure 5: Pivot results: The hybrid Coral/C++ group (avg. 2.4 / 2.2adj.) pivots more than the pure C++ group (avg. 1.3 / 1.5adj.) each week.

Figure 5 shows that the hybrid Coral/C++ group (2.4) pivots more frequently each week than the pure C++ group (1.3). Even after removing week 6 from the calculations to account for the midterm, the hybrid Coral/C++ group (2.2) still pivots more than the pure C++ group (1.5).

5 Conclusion

In this paper, we shared our experience using a hybrid Coral/C++ MSP teaching approach in our CS1 class. We found that using a hybrid Coral/C++ approach did not harm student grade performance. We found that both groups spent a healthy amount of time working on lab activities. We saw that students in the hybrid group developed their code more and submitted their code less frequently than the pure C++ group. Both groups start working about 4 days before the deadline and both groups make good use of pivoting. This work is not meant to conclude that one teaching approach is better, but rather to show that both approaches work. Using a Coral/C++ approach to begin a CS1 class does not harm students but can offer benefits such as having an easier time teaching programming fundamentals when the class begins. As such, we will likely continue using this approach in our CS1, and we encourage others to try this approach as well.

References

[1] P. Kinnunen and L. Malmi, "Why Students Drop Out CS1 Course?" Proceedings of the second international workshop on Computing education research. 2006.

[2] J. Bennedsen and M. E. Casperson, "Failure rates in introductory programming: 12 years later," in ACM Inroads, 2019.

[3] S. Bergin and R. G. Reilly, "The Influence of Motivation and Comfort-Level on Learning to Program," in PPIG, 2005.

[4] C. Watson and F. W. B. Li, "Failure rates in introductory programming revisited," in Proceedings of the 2014 conference on Innovation technology in computer science education (ITiC).

[5] J.M. Allen, F. Vahid, A. Edgcomb, K. Downey, and K. Miller, “An Analysis of Using Many Small Programs in CS1,” ACM SIGCSE Technical Symposium on Computer Science Education, 2019.

[6] J.M. Allen, F. Vahid, K. Downey, and A. Edgcomb, “Weekly Programs in a CS1 Class: Experiences with Auto-graded Many-small Programs (MSP),” Proceedings of ASEE Annual Conference, 2018.

[7] Coral. https://corallanguage.org/ Accessed: August, 2020.

[8] A. D. Edgcomb, F. Vahid, and R. Lysecky. "Coral: An ultra-simple language for learning to program." ASEE Annual Conference and Exposition, Conference Proceedings. 2019.

[9] Snap. https://snap.berkeley.edu/ Accessed: August, 2020.

[10] Scratch. https://scratch.mit.edu/ Accessed: August, 2020.