



Improving Pass Rates by Switching from a Passive to an Active Learning Textbook in CS0

Ms. Dawn McKinney, University of South Alabama

Dawn McKinney, a Senior Instructor and Curriculum Coordinator for Computer Science at the University of South Alabama, has been conducting research on Teaching and Learning for over 23 years and has co-authored over 25 papers which have been presented at SISCSE, ASEE, FIE, XP/Agile Universe, International Conference on The First-Year Experience, Southeastern Learning Community Consortium, Council on Undergraduate Research National Conference, and the South Alabama Conference on Teaching and Learning. As a leader in the university's Team-Based Learning effort, McKinney has been awarded funds for support, including travel, for the past seven years. She taught courses in China in 2013 and was awarded the highest award for teaching at the University of South Alabama in 2014. During the last three years, McKinney has participated in the Scholarship on Teaching and Learning program supported by the University of South Alabama and has been awarded funds to use for travel. During this time McKinney has collaborated with computer science faculty at several institutions and has co-authored papers submitted to both SIGCSE and ASEE.

Dr. Alex Daniel Edgcomb, Zybooks

Alex Edgcomb is a Senior Software Engineer at zyBooks.com, a startup spun-off from UC Riverside that develops interactive, web-native learning materials for STEM courses. Alex is also a research specialist at UC Riverside, studying the efficacy of web-native content and digital education.

Prof. Roman Lysecky, The University of Arizona

Roman Lysecky is a Professor of Electrical and Computer Engineering at the University of Arizona. He received his Ph.D. in Computer Science from the University of California, Riverside in 2005. His research focuses on embedded systems with emphasis on medical device security, automated threat detection and mitigation, runtime adaptable systems, performance and energy optimization, and non-intrusive observation methods. He is an author on more than 100 research publications in top journals and conferences. He received the Outstanding Ph.D. Dissertation Award from the European Design and Automation Association (EDAA) in 2006, a CAREER award from the National Science Foundation in 2009, and seven Best Paper Awards. He is an inventor on one US patent. He has authored eight textbooks on topics including C, C++, Java, Data Structures, VHDL, and Verilog, and he has contributed to several more. His recent textbooks with zyBooks utilize a web-native, active-learning approach that has shown measurable increases in student learning and course grades. He has also received multiple awards for Excellence at the Student Interface from the College of Engineering at the University of Arizona.

Prof. Frank Vahid, University of California, Riverside

Frank Vahid is a Professor of Computer Science and Engineering at the Univ. of California, Riverside. His research interests include embedded systems design, and engineering education. He is a co-founder of zyBooks.com.

Improving Pass Rates by Switching from a Passive to an Active Learning Textbook in CS0

Abstract

Undergraduate degree programs in computer science have struggled with student retention and outcomes; some solutions focus on improving introductory courses, such as CS0. This paper reports on one such approach: Switching from a passive to active learning textbook, which uses animations, learning questions, and simulators, along with text and figures. The Fall 2017 offering had 73 students and used a passive textbook and a standalone flowchart simulator. The Fall 2018 offering had 76 students and used an active learning textbook, including integrated auto-graded homeworks and integrated flowchart simulator. The primary change in the course was to the textbook; the courses had the same experienced instructor; similar quizzes, exams, and homeworks; and similarly-prepared students based on ACT scores and previous programming experience. Various metrics were measured; most crucially, the pass rate significantly increased (p -value = 0.04) from 78% (57 of 73 students) in Fall 2017 to 91% (69 of 76 students) in Fall 2018. In Fall 2017, 10 of the 16 students that did not pass changed majors, whereas only 2 of the 7 did in Fall 2018. The course grades increased from 84 out of 100 points in Fall 2017 to 87 in Fall 2018; the largest categorical increase (p -value < 0.001) was homework from 71 out of 100 points in Fall 2017 to 88 in Fall 2018. Students were surveyed about the course; significant findings turned out to be that Fall 2018 students found the homework to be clearer but harder than Fall 2017 students.

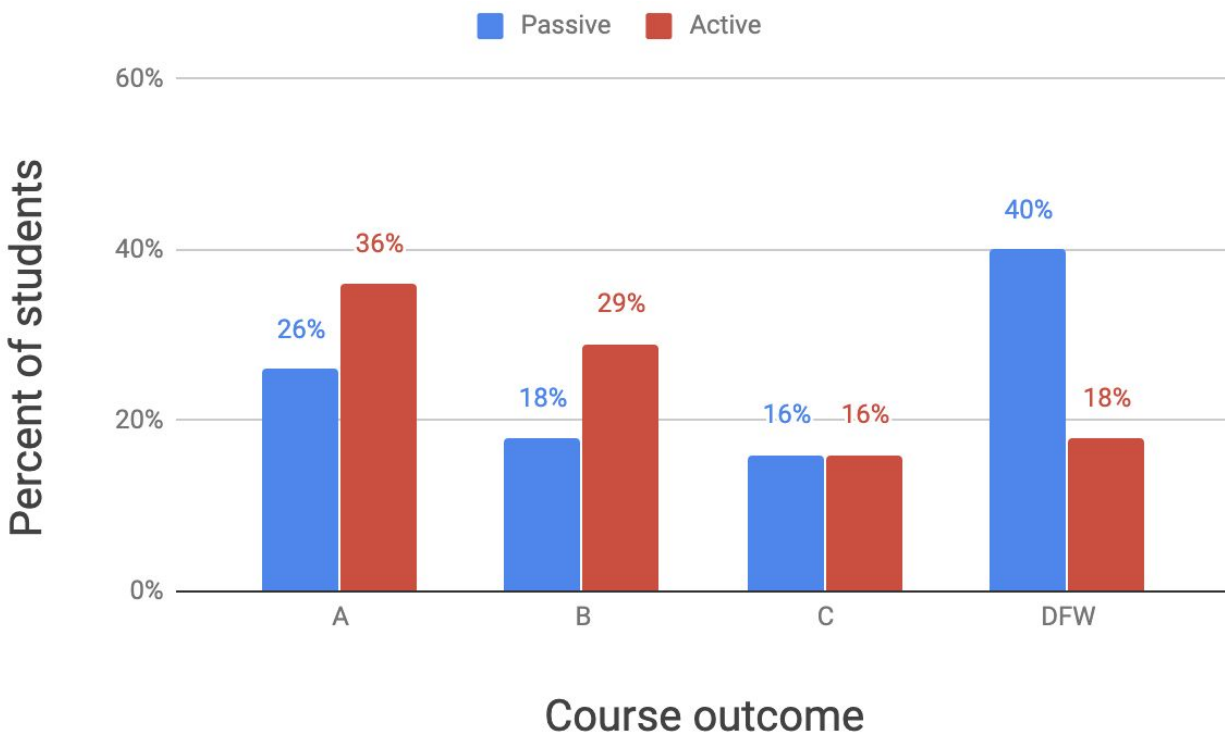
1. Introduction

Students struggle in introductory programming courses, often due to lack of good problem-solving skills and lack of preparation. Many institutions have implemented and adjusted courses to address this deficiency [1] - [4]. The CS Department in the School of Computing at the University of South Alabama has been searching for an appropriate first course for CS majors to prepare them for the programming courses. In response to observed student struggle with problem-solving and programming concepts, the department decided to focus on such skills in the first course. In Fall 2016, the designers of the course used a free tool, which allowed students to create runnable flowcharts to test their algorithms and an accompanying textbook based on the same tool. The students enjoyed the use of the tool, but the instructor and a teaching assistant had to grade each assignment by hand because there was no feedback mechanism integrated into the tool. With 60 students, providing students with high quality, timely feedback across many assignments was not feasible. So, only a few assignments were graded, typically with a one-week turnaround. Furthermore, the tool was designed for non-majors, and students

reported dissatisfaction with the low level of feedback for their efforts, even though the tool does a good job of visually demonstrating algorithms with flowcharts. Without timely feedback, students felt low incentive to practice. The class was not able to progress deep enough into programming concepts, and the students did not do as well as expected on exams and homework.

The course designers searched for an alternative. The CS1 instructor was dissatisfied with the passive learning textbook in the CS1 course that teaches Java. The textbook was expensive and seemed to be rarely used by students. Students were not getting the feedback needed on homework to improve their programming skills and were not well prepared for CS2. During the Fall 2017 semester, an active learning textbook was used in the CS1 course. The instructors of the course compared the success rates and attitudes of students before and after the use of the active learning textbook in CS1 [5]. As shown in Figure 1, the semester using the active learning textbook had more As and Bs, and fewer DFWs than the semesters using the passive learning textbook. Course evaluations for the active learning textbook version of the CS1 course indicated the students were satisfied with the textbook.

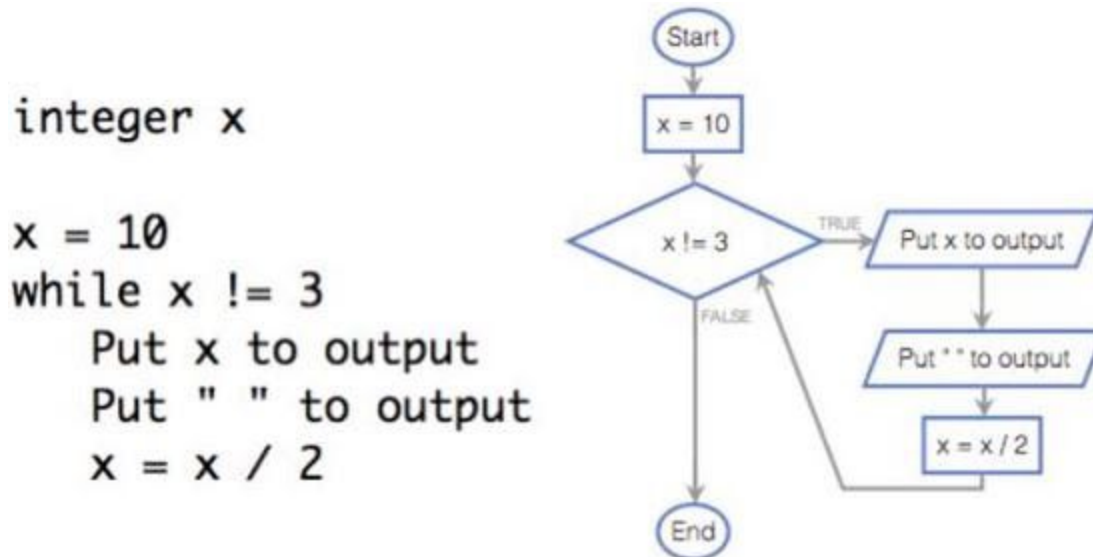
Figure 1: CS 1 (Java) student outcomes of the last 4 semesters with passive learning textbook (156 students total) and first two semesters with active learning textbook (81 students).



The success in CS1 led to the attempt to use an active learning textbook in CS0, which is the focus of this paper. In Spring 2018, the CS0 instructor adopted an active learning textbook,

which teaches programming concepts using a simple pseudocode language with corresponding flowcharts, shown in Figure 2. The active textbook provides several benefits: (1) students get immediate feedback on ample assignments and activities, (2) a single source for learning and practice material, (3) active learning and visual material, (4) low-cost, (5) the instructor is provided with reports that can be used for grades.

Figure 2: Pseudocode and corresponding flowchart. The code on the left defines the same program as the flowchart on the right.



This paper compares a passive textbook and an active learning textbook in the context of a CS0 course across two semesters during the same term of the year (Fall 2017 and Fall 2018), with the same experienced instructor, and all other course items kept as similar as possible. The paper analyzes the effects on student outcomes and perspectives. The paper begins with the background work, gives a course overview, then describes the change in the course (the textbook), participants, design and methods, results, and discussion.

2. Background

Introductory programming courses commonly struggle with low student success rates [6], [7]. Math has long been a determining factor for student success [8], [9], but others have found that this is not always the case [10]. Causes for low student success rates have been identified which range from time-management and ability, to commitment level and motivation [11] - [13]. Many approaches have been taken by universities, including ours, to remedy this issue, such as changing programming languages [14], focusing on student motivation [15] - [17], implementing team-based learning [18] - [21], adopting Agile methodologies [22], and a host of others.

At our university, we discovered that students were not benefitting from the textbook, and were not getting enough practice and timely feedback to satisfy their needs for success. Further, students expect more use of technology and the Internet and are not interested in expensive, heavy, and lengthy textbooks. During Fall 2017, we found an active-learning textbook to replace a static textbook for our CS1 course which is completely online, inexpensive, and provides practice and immediate feedback for the students. We compared student success rates over several years to the success rates after adopting the new active-learning experience and the encouraging results [5] led to the adoption of the experience for our CS0 course the following semester (Spring 2018).

We collected preliminary data in Spring 2018 and made a few improvements in preparation for this study, which compares student success rates in the CS0 course from Fall 2017 (traditional passive textbook experience) to Fall 2018 (active-learning textbook experience) and will show that the active-learning textbook experience resulted in improved student success and attitudes about the course.

3. Course overview

The course in this study was an Introduction to CS (CS0), which was required for CS majors and 3 semester-hours, and used Team-Based Learning (TBL). Fall 2017 had 73 students and Fall 2018 had 76 students. The course focused mostly on problem-solving and the algorithmic process, and was used as a ramp for the next course (CS1), which is the first programming course. High ACT Math students are permitted to take both CS0 and CS1 courses concurrently since the introductory course (CS0) also includes a brief computer history, data representation, binary numbers, speakers from the CS department who share their research, and the students explore their interests in CS by completing a semester project which culminates with a team presentation.

To develop programming problem-solving skills, students practice writing pseudocode and using flowcharts to solve simple problems which make use of the basic programming constructs, e.g., sequential, decision, and repetition structures. Programming concepts such as identifiers, data types, variables, assignments, arithmetic operations, relational and logical operators, and functions are introduced and used by students to complete exercises.

3.1 Team-based learning

Team-Based Learning (TBL) [20] is a pedagogy adopted by the University of South Alabama as an initiative for improving teaching and learning. Students in TBL courses are assigned to permanent teams between five and seven in size. Four to six major topics are covered with

accompanying reading and assignments that are given in preparation for short multiple-choice quizzes for each topic. The quiz, known as a RAT (Readiness Assurance Test) is first taken by each individual (iRAT) and immediately followed by completing the same quiz by the entire team (tRAT). In our course, the iRAT is completed online and an immediate feedback tool is used for the tRAT. Each of the 4-6 RATs is followed by discussion and clarification of concepts. This process is known as a RAP (Readiness Assurance Process). Following a RAP, teams work together on in-class activities on each major topic and evaluate their team members [21].

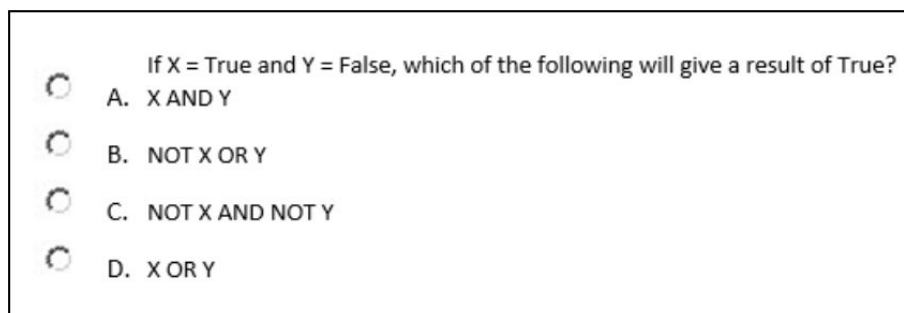
3.2 Grades

Grades in the course were based on participation and attendance (5%), individual readiness assurance tests (10%), homework (15%), exams (25%), team project (10%), team evaluations (15%), and team readiness assurance tests (20%).

3.2.1 Individual readiness assurance tests

Individual Readiness Assurance Tests (iRATs) are short tests (under 10 minutes; 10 multiple choice questions) given at the beginning of the class period when homework (reading with practice) is due. iRATs are designed to be a quick check to make sure students are doing the homework in order to be prepared for the team activities. Figure 3 is an example iRAT question involving logical operators. The iRATs are given in an online environment so the instructor has immediate feedback about how well the students did. Immediately following the iRAT, the tRAT uses scratch off forms so teams know if their answers are correct.

Figure 3: Example Readiness Assurance Test (RAT) question.



If $X = \text{True}$ and $Y = \text{False}$, which of the following will give a result of True?

- A. $X \text{ AND } Y$
- B. $\text{NOT } X \text{ OR } Y$
- C. $\text{NOT } X \text{ AND NOT } Y$
- D. $X \text{ OR } Y$

3.2.2 Individual homework

Homework was assigned in preparation for each of the 5 major topics. Each major topic covers 1-3 chapters. In Fall 2017, students were assigned reading from a textbook and were required to complete a few short exercises (which had answers available) in order to test their understanding

of the reading. There was not a way to verify that students read the textbook, and the textbook's exercises were not graded. However, since a RAT was given on the due date of homework assignments, the students did have an incentive to read.

Five graded homework challenges were also given, which, with the passive learning textbook, involved the use of a free flowcharting tool that was supported by the textbook. Students completed the flowcharts and uploaded their work to an online system where a grader manually graded each student's work using a rubric provided by the instructor. The rubric included 4 categories: Readable and understandable, correct flow of control, correct use of identifiers, and correct output. Each category had 4 tiers: Good (25 points), fair (20 points), poor (15 points), and missing (0 points). For readable and understandable, a good answer was a solution with three components: An algorithm that was readable, understandable, and using indentation where needed. A fair was missing one component; poor was missing two components; and missing had no components. Each category followed a similar pattern.

In Fall 2018, the same five graded challenges were given, but the students used a Coral simulator instead of the flowcharting tool. The simulator gave the students immediate feedback for their efforts allowing students to make corrections. Since the simulator required proper indentation and some rules for syntax, the students learned to be exact in the design of their solutions. Exact output was also required, which further reinforced the importance of the specificity of problem-solving and programming. The active learning textbook was introduced to the course in the Spring 2018 offering, between the offerings compared in this paper, Fall 2017 and Fall 2018.

3.2.3 Exams

A midterm (students allotted 65 minutes) and final exam (students allotted 120 minutes) were given. The exams included multiple choice (35 questions on the midterm; 45 on the final along with team evaluations). In both Fall 2017 and Fall 2018, the midterm also included a free response question.

4. Changes in the course

The only change in the course offerings was the textbook and associated simulator. Everything else stayed the same: Grading policies, course point breakdown, topics, number of exams, assignments, term of the year, class size, and experienced instructor. In Fall 2017, a traditional print textbook was used, along with a flowchart building tool. In Fall 2018, an active learning textbook with an integrated simulator was used.

The passive learning textbook was *Prelude to Programming, Concepts and Design, Sixth Edition* [23]. The textbook used flowchart diagrams and textual explanations to describe core programming concepts. The textbook was accompanied by a standalone flowchart interpreter, RAPTOR [24]. The textbook had exercises and problems at the end of each chapter and demonstrations and examples on how to use the interpreter. The interpreter enables a user to build a flowchart by inserting, connecting, and editing nodes. The user can layout the nodes however desired. The textbook included explicit instructions for how to use the interpreter, as well as how to build particular flowcharts as examples.

The active learning textbook was *Fundamental Programming Concepts* [25]. The textbook included animations to show dynamic concepts, learning questions with explanations, and tools with immediate feedback, as well as, text and figures. Such active learning textbooks have been shown to improve student learning in introductory programming courses [26], [27]. Student completion of activity is recorded, and instructors can see that completion. The textbook introduces basic programming concepts, including representing data as bits, problem solving, and programming. The textbook introduces programming concepts via the Coral programming language [28]. The language is a textual and flowchart language (shown in Figure 2), wherein a user writes textual code that is auto-generated into a flowchart. The user can visually see the textual code and flowchart executing, as well as see variables update in memory. The language's syntax is very simple, supporting only constructs for basic programming concepts: Input/output, variables, branching, loops, arrays, and functions.

The passive learning textbook had homework in a separate document; homework was hand-graded. The active learning textbook had the homeworks integrated into an auto-grader with immediate feedback for students.

5. Participants

The participants were students in the Fall 2017 and Fall 2018 offering of an introductory programming course at a research university. The Fall 2017 offering had 73 students, and the Fall 2018 offering had 76 students. Most participants were freshmen, Computer Science majors, and traditional students with an occasional transfer or non-traditional student.

To get a sense for the student's level of preparation, we looked at ACT Math scores (taken prior to entry into the university; most students were freshmen) and survey questions. The average ACT Math score for Fall 2017 students was 23.6, which was not significantly higher (p -value = 0.48) than the average score for Fall 2018 students at 23.0. Participants were given a survey at the end of the third homework with 3 questions related to prior programming experience. 57% of Fall 2017 participants vs 28% Fall 2018 (p -value = 0.02) reported having taken a prior

programming course. On a scale of 0 (no experience) to 5 (a lot of experience), Fall 2017 students reported 1.3 vs Fall 2018 at 0.9 (p-value = 0.23). 50% of Fall 2017 students reported taking a concurrent programming course vs 49% of Fall 2018 students (p-value = 0.91). The offerings seem to have similar levels of prepared students.

6. Design and methods

The two course offerings were compared via student outcomes and perspectives. Including:

- Pass rate: The number of students who received a course grade of A, B, or C divided by the number of students that were enrolled.
- Course grades: Overall course grade and each grade category: Homework, attendance, quizzes, midterm exam, and final exam. Students who did not take the final were excluded.
- Survey responses: The same survey was given after the 3rd, 4th, and 5th homework (there were 5 homeworks). The first survey question asks how long the student spent on homework; the other questions were on the scale of -3 (strongly disagree) to 3 (strongly agree). An average question score was computed across all surveys by first averaging each student's response per survey, then combining the 3 surveys per offering. For significance testing, survey responses were combined by Z-scoring each survey per question, then concatenating the question Z-scores.
- Course evaluation responses: Both offerings had the same course evaluation given at the end of the semester. 4 items on the course evaluation were related to the change in textbook.

For pass rate, significance was computed with Fisher's exact test.

For course grades and survey responses, Bartlett's test was performed for each metric. If Bartlett's test yielded p-value < 0.05, then variances were considered unequal and the significance value between offerings was computed via Welch's t-test. Otherwise, variances were considered equal, and the significance value was computed via Student's t-test. Either way, significance value was computed with two-tails. Further, a Bonferroni correction was applied to the interpretation of the significance value due to multiple tests. For course grades, a p-value < 0.008 (= 0.05 / 6 tests) is interpreted as statistically significant. For survey responses, a p-value < 0.005 (= 0.05 / 10 tests) is interpreted as significant.

R was used for all analysis [29]. The Internal Review Board approved this study.

7. Results

The pass rate in Fall 2017 (passive learning textbook) was 78% (57 of 73 students), which was significantly lower (p-value = 0.04) than 91% (69 of 76 students) in Fall 2018 (active learning

textbook). Of the students who did not pass, 10 changed majors in Fall 2017, and 2 changed majors in Fall 2018.

Course grades are shown in Table 1. Fall 2018's homework average was 88 out of 100 points, which is interpreted as significantly higher ($p\text{-value} < 0.001$) than Fall 2017's homework average of 71. Of the grade categories, homework most utilized the textbooks and respective simulators. The other grade categories are not interpreted as significantly different because the p -values are not less than the Bonferroni correction of 0.008. The offerings had unequal variances for homework and attendance, and equal variances for quizzes, midterms, finals, and overall. Note that students who did not take the final (such as dropped students) were not included in the grades comparisons.

Table 1: Course grades, including each category, for Fall 2017 and 2018. Homework was significantly different, so is bold. To address p -hacking, we used a Bonferroni correction, so instead of significance being < 0.05 , significance must be < 0.008 .

	N	Homework*	Attendance*	Quizzes	Midterm	Final	Overall
Fall 2017	62	71%	94%	75%	81%	84%	84%
Fall 2018	70	88%	90%	70%	79%	78%	87%
	p-value	< 0.001	0.08	0.25	0.22	0.02	0.05

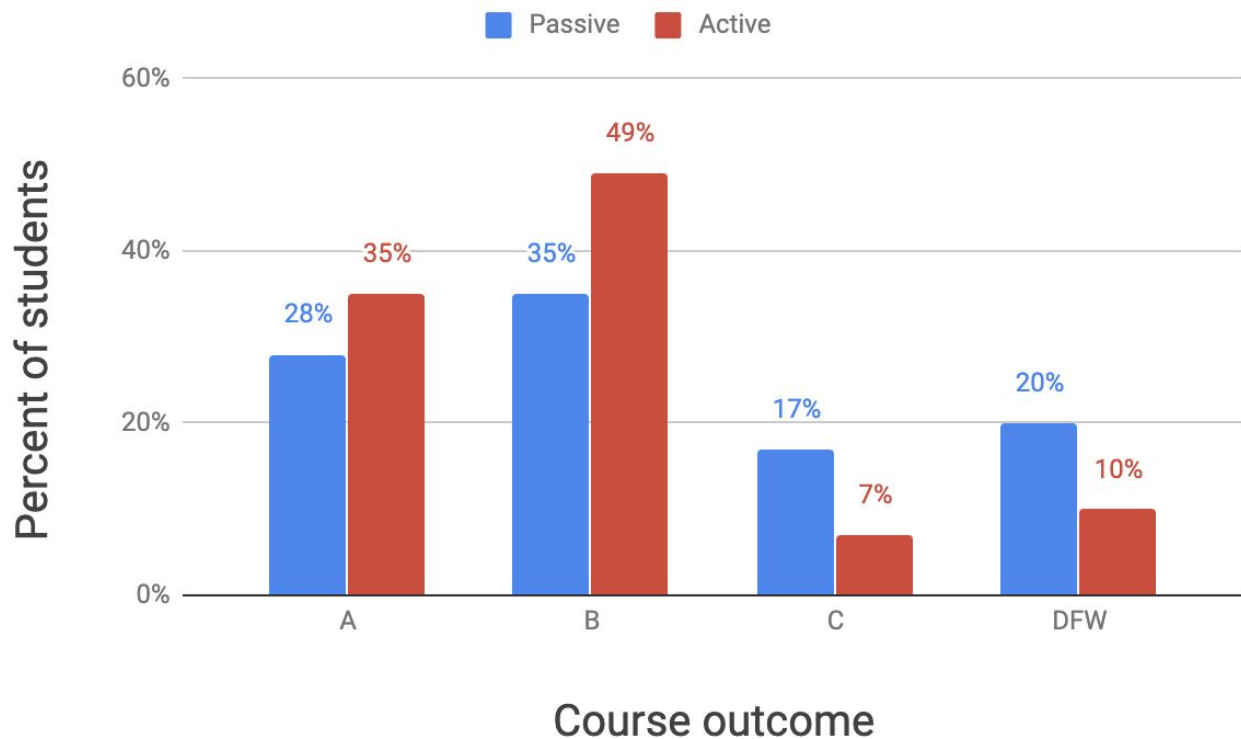
* indicates significance test for unequal variance used.

A note regarding Table 1: "p-hacking" is a growing concern among researchers today. To be clear, the data presented in this paper is not p-hacked. The paper presents all the measured items, not just those of significance. Furthermore, the paper uses a standard technique, called a Bonferroni correction, that imposes a higher standard on significance due to multiple items being presented (versus a hypothesis with a single measured item). In particular, p must be 0.001, not just 0.05, to be considered significant.

Some Finals' questions covered concepts not discussed in both offering's textbooks; in fact, some questions in the Fall 2018 Final exam asked about terms not used in the Fall 2018 textbook (rather, these questions were vestiges from Fall 2017). When such questions were excluded, Fall 2017 students averaged a 78, and Fall 2018 students averaged 87.

As seen in Figure 4, Fall 2018 course grades had more As and Bs, and fewer Cs and DFWs than Fall 2017.

Figure 4: CS0 student outcomes with a passive (Fall 2017) and active (Fall 2018) learning textbooks.



Survey questions are shown in Table 2. Fall 2018 (active learning textbook) students responded that homework instructions were clearer (0.9 or slightly agree vs 1.8 or agree) than Fall 2017 (passive learning textbook) students responded, which is significant because the p-value of 0.003 is less than Bonferroni correction of 0.005. Fall 2018 students reported the homework being more difficult (0.4 vs 2.0) than Fall 2017 students reported, which was significant (p-value < 0.001). Fall 2018 students reported having more difficulty writing the program for the solution thought of (0.2 vs 1.1) than Fall 2017 students reported, which was significant (p-value = 0.001).

The course evaluation items were on a scale from 1 (strongly disagree) to 5 (strongly agree). The items were: Useful feedback provided (3.7 Fall 2017 vs 3.9 Fall 2018; p-value = 0.29), exams/assignments match course material (4.1 vs 4.2; p-value = 0.71), assigned materials appropriate (4.1 vs 4.1; p-value = 0.91), and course overall (3.8 vs 3.9; p-value = 0.45). No item was significantly different.

Table 2: Survey questions with average scores. With Bonferroni correction, significance is p-value < 0.005. Except first question, scale of -3 (strongly disagree) to 3 (strongly agree) used.

Question	Fall 17 average	Fall 18 average	p-value
In total, how long did the homework take to complete? (hours)	2.5	1.8	0.288
I enjoyed the assignment.*	1.0	1.1	0.647
The homework was an appropriate amount of work.	1.5	1.8	0.596
I was able to solve the homework.*	1.8	1.6	0.174
The homework instructions were clear.*	0.9	1.8	0.003
The homework was difficult.*	0.4	2.0	< 0.001
The textbook contributed to my success on the homework.	0.6	1.1	0.471
I felt prepared for the homework.	1.1	0.9	0.127
I encountered a lot of program bugs while solving the homework.	0.4	0.1	0.706
I had a hard time writing the program to match the solution I thought of.	0.2	1.1	0.001

* indicates significance test for unequal variance used.

8. Discussion

This paper compared passive and active learning textbooks in the context of a CS0 course across two semesters. The paper analyzed the effects on student outcomes and perspectives. Overall, our results suggest that students using an active learning textbook had significant increases in pass rates. These findings have important implications for CS instructors of introductory courses and CS departments, who are trying to improve pass rates and retention. This paper is the first to compare passive and active learning textbooks in a CS0 course that focuses on teaching programming concepts via pseudocode and flowcharts. The results are consistent with past literature comparing passive and active learning textbooks in CS1 and other contexts.

The pass rate increased significantly from 78% to 91% after switching to an active learning textbook. Such pass rates likely have implications for retention, as evidenced by an analysis of students who failed the course: Only 2 of 7 students (29%) changed majors after the active learning textbook course, whereas 10 of 16 students (63%) changed majors after the passive learning textbook course. Such improvements are encouraging, as many CS departments are

struggling with retention issues, including the authors' own CS departments. Simply changing to an active learning textbook is not sufficient to solve retention issues; there is no silver bullet in education, but such a change may be consistent with department goals.

Course grades improved modestly, with large improvements in the homework scores and modest decreases in the exam scores. The Fall 2018 final exam included some questions on terms not used in the Fall 2017 textbook. Also, both final exams included questions on concepts not covered in both textbooks. When such questions were excluded, Fall 2017 students averaged a 78, and Fall 2018 students averaged 87. Fall 2018 was just the second offering after switching textbooks, so some kinks were still being worked out, whereas multiple terms had used the same textbook prior to Fall 2017, thus presumably more kinks were worked out.

Students who did not take the final exam were excluded from the grade analysis. Such students tend to have dropped the course or already had a failing grade (so no point in taking the final exam). However, Fall 2018 had more passing students, so more students who may have decided not to take the final exam in 2017 instead did take the Final in 2018. This is a critical point for instructors to expect: When fewer students drop the course, overall grades are likely to decrease because the students who would have dropped, but did not, are likely to have lower than average course grades.

With the active learning textbook, homework grades increased significantly, while students reported less time spent and the homework being more difficult. Both offerings had the same homework by the instructor and students used the flowchart simulator associated with the respective textbooks. However, the active learning homework was auto-graded with immediate feedback, whereas the passive learning homework was hand-graded with delayed feedback (about a week). Auto-grading let the student know the current score immediately, so students may have felt encouraged to continue trying, which may explain both the increased homework grade and difficulty. Also, students wrote pseudocode to solve the active learning homeworks, whereas students directly built a flowchart for the passive learning homeworks. Building a flowchart may actually be easier or more intuitive for new CS students than writing pseudocode, as evidenced by the proliferation of block-based and flow-based languages for younger students. However, industry languages are text-based, so having CS students write pseudocode may be beneficial for student outcomes in subsequent courses, like CS1, and retention in the long run.

9. Limitations and future work

This paper describes a cross-semester naturalistic study of an introductory CS course. In such analyses, controlling potentially confounding factors is hard. For example, students are not randomly assigned to course offerings, but rather assigned by the term that the student chose to

take the course. However, such factors are likely mitigated by comparing two offerings of the same course at the same university, exactly one year from each other with the same experienced instructor who modified a few course items. A future study might include multiple courses from different universities to help mitigate confounds.

There are many differences between passive and active learning materials, which makes pinpointing the specific cause of improvement challenging. For example, the difference in perceived homework difficulty may have been caused by an auto-grader vs hand-grader, or immediate feedback vs delayed feedback, or writing pseudocode vs building a flowchart, and so on, or just the fact the textbook changed.

The survey helped identify that the difficulty was with writing the solution (rather than thinking of the solution). However, more might be learned about this difficulty. For example, a future survey might include a question like: I had a hard time coming up with a solution to the homework.

An interesting aside: We adjusted CS0 and CS1 enrollment policy in Fall 2019 to only allow students to take one course at a time, so Fall 2019 CS0 only consisted of underprepared students. This adjustment improved student perceptions of CS0, and enabled the instructor to focus specifically on problem-solving, which is the core skill taught in our CS0. Fall 2019 CS0 students again reported satisfaction with the active learning textbook.

10. Conclusion

In a CS0 course teaching core programming concepts via pseudocode and flowcharts, switching from a passive (73 students) to active learning textbook (76 students) resulted in increased pass rates (78% to 91%; p -value = 0.04) and fewer changes out of the CS major for non-passing students (10 students vs 2 students). Overall course grades increased modestly, but homework grades increased significantly (71% to 88%; p -value < 0.001). Interestingly, students with the active learning textbook reported the homeworks being significantly more difficult, but the students actually spent a bit less time on homework and felt the instructions were much clearer, which makes sense because immediate grading feedback makes clear that one's program has mistakes. Such findings are relevant to CS instructors and departments, as many universities struggle to increase retention and pass rates in lower-division CS courses, particularly introductory courses.

References

- [1] Reynolds, J., R. Adams, R. Ferguson, and P. Leidig. Programming in the IS Curriculum: Are Requirements Changing for the Right Reason? *Information Systems Education Journal*, 15(1), 80, 2017.
- [2] Rolka, C. and A. Remshagen. Showing Up Is Half the Battle: Assessing Different Contextualized Learning Tools to Increase the Performance in Introductory Computer Science Courses. *International Journal for the Scholarship of Teaching and Learning*, 9(1), n1, 2015.
- [3] Uysal, M.P. Improving First Computer Programming Experiences: The Case of Adapting a Web-Supported and Well-Structured Problem-Solving Method to a Traditional Course. *Contemporary Educational Technology*, 5(3), 198-217, 2014.
- [4] Yagci, M. Blended Learning Experience in a Programming Language Course and the Effect of the Thinking Styles of the Students on Success and Motivation. *Turkish Online Journal of Educational Technology-TOJET* 15, no. 4: 32-45, 2016.
- [5] Clark, G. and D. McKinney. The Impact of an Innovative Learning Environment for a Programming Course. Poster presented at the Eighth Annual South Alabama Conference on Teaching and Learning. Mobile, AL, 2018.
- [6] Bennedsen, J. and M. Caspersen. Failure rate in introductory programming. *ACM SIGCSE Bulletin*, 39(2), 32-36, 2007.
- [7] Watson, C. and F. Li. Failure rates in introductory programming revisited. *ITiCSE '14 Proceedings of the Conference on Innovation and Technology in Computer Science Education*, 39-44, 2014.
- [8] Jordan, K. and G. Stein. The Math Gap in an Inclusive CS1 Course. *Proceedings of the 49th SIGCSE Technical Symposium on Computer Science Education*, 2018.
- [9] Wilson, B. and S. Shrock. Contributing to success in an introductory computer science course: a study of twelve factors. *Proceedings of the 32nd SIGCSE Technical Symposium on Computer Science Education*, 2001.
- [10] Boyle, R., J. Carter, and M. Clark. What makes them succeed? Entry, progression and graduation in computer science. *Journal of Further and Higher Education*, 26(1):3–18, 2002.
- [11] Corney, M., D. Teague, and R.N. Thomas. Engaging students in programming. *Twelfth Australasian Conference on Computing Education*. Volume 103. 63–72, 2010.
- [12] Nikula, U., O. Gotel, and J. Kasurinen. A motivation guided holistic rehabilitation of the first programming course. *ACM Transactions on Computing Education*, 2011. DOI: 10.1145/2048931.2048935
- [13] Teague, D. and P. Roe. Collaborative Learning – towards a solution for novice programmers. In: *Tenth Australian Computing Education Conference*. 147–154, 2008.
- [14] Murphy, E., T. Crick, and J.H. Davenport. An Analysis of Introductory Programming Courses at UK Universities. *The Art, Science, and Engineering of Programming* 1.2: 18-1, 2017.

- [15] Denton, L.F., D. McKinney, and M.V. Doran. Promoting Student Achievement with Integrated Affective Objectives. Proceedings of ASEE Annual Conference, 2003.
- [16] McKinney, D. and L.F. Denton. Houston, we have a problem: there's a leak in the CS1 affective oxygen tank. Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education, 2004.
- [17] McKinney, D. and L.F. Denton. Affective Assessment of Team Skills in Agile CS1 Labs: The Good, the Bad, and the Ugly, Proceedings of the 36th SIGCSE Technical Symposium on Computer Science Education, 2005.
- [18] McKinney, D. Where do I belong: A team-based, inquiry-based, and service-learning approach to an introductory course in computer science. Sixth Annual South Alabama Conference on Teaching and Learning. Mobile, AL, 2016.
- [19] McKinney, D. and L.F. Denton. Developing Collaborative Skills Early in the CS Curriculum in a Laboratory Environment, Proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education, 2006.
- [20] Michaelsen, L. K., A.B. Knight, and L.D. Fink. Team-based learning: A transformative use of small groups in college teaching (1st pbk. ed.). Sterling, VA: Stylus P, 2004.
- [21] TBL. <http://www.teambasedlearning.org/definition/>. Accessed: April 2020.
- [22] McKinney, D., J. Froeseth, J. Robertson, L.F. Denton, and D. Ensminger. Agile CS1 Labs: eXtreme Programming Practices in an Introductory Programming Course. Conference on Extreme Programming and Agile Methods. Springer, Berlin, Heidelberg, 2004.
- [23] Venit, S. and E. Drake. Prelude to Programming, Concepts and Design, Sixth Edition. Pearson, 2015.
- [24] RAPTOR - Flowchart interpreter. <https://raptor.martincarlisle.com/>. Accessed: April 2020.
- [25] zyBooks. <http://www.zybooks.com/>. Accessed: April 2020.
- [26] Edgcomb, A., F. Vahid. Effectiveness of Online Textbooks vs. Interactive Web-Native Content, Proceedings of ASEE Annual Conference, 2014.
- [27] Edgcomb, A., F. Vahid, R. Lysecky, A. Knoesen, R. Amirtharajah, and M.L. Dorf. Student Performance Improvement using Interactive Textbooks: A Three-University Cross-Semester Analysis, Proceedings of ASEE Annual Conference, 2015.
- [28] Coral. <http://corallanguage.org/>. Accessed: April 2020.
- [29] R Core Team. R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria, 2019.