Evolving a Data Structures Class Toward Inclusive Success

Celine Latulipe, Stephen MacNeil, Brian Thompson
Department of Software and Information Systems
University of North Carolina at Charlotte
Charlotte, NC, USA
{clatulip, smacnei2, bthomp57}@uncc.edu

***Abstract*—This paper presents the evolution of a curricular design over four semesters with the intention of improving learning for all while increasing retention and equity in a fully flipped, team-based data structures class. Using peer instruction, lightweight teams, an automated grader, TA-mentorship, and pair programming labs, we create an inclusive, equity-driven peer learning environment. We describe our holistic and iterative course design, reports and analysis of student background and performance, results from anonymous student surveys, and future plans for further improving the inclusive success in computing courses.**

***Index Terms*—data structures, lightweight teams; flipped classes; inclusion, equity**

I. Introduction

Enrollment in CS courses over the last few years has been described as a wave and as a tsunami [1], [2]. This increase in enrollment stretches teaching resources and leads to large classes and impersonal learning environments where students just attend lectures, do homework, and take tests. Such learning environments have been shown to lead to competition over cooperation and superiority over empathy [3]. Students with prior experience in computing tend to outperform their less-prepared peers, and are less likely to share their knowledge, which further stratifies inequalities between students with differing levels of prior experience. These learning environments propagate negative stereotypes about CS such as it being competitive, singularly-focused, asocial, and primarily male [4]. These perceptions and experiences are important because students who do not have a sense of belonging and community in their major tend to doubt their place in their future field [5]. To address these issues, we have iteratively developed an inclusive pedagogy for data structures, a core computing course, to encourage a sense of community and cooperative learning success.

Our institution is an urban research university in the US, and qualifies as a minority-serving institution, with more than 25% of first-year students identifying as African-American or Latinx. The data structures course is a critical juncture in students' educational journeys where we lose a high proportion of students, particularly under-represented students.

The authors acknowledge partial funding for the research reported in this paper: NSF Award 1519160: IUSE/PFE:RED: The Connected Learner: Design Patterns for Transforming Computing and Informatics Education.

One approach to increase student success and retention in computing courses is the use of lightweight teams and flipped classes [6]. Eddy and Hogan have shown initial evidence that the higher levels of structure in flipped classes increase success for minority and first generation college students [7]. Lightweight teams are rooted in pedagogical techniques of active learning and team-based instruction in flipped classes that continue to demonstrate success across many domains, as evidenced by a meta-analysis of 225 studies in STEM [8]. Many studies of active learning focus on introductory classes since these are often students' first experiences with computing [9], [10], and some have shown that active learning in CS1 classes can lead to better performance and retention in later classes [11]. Lightweight teams consistently show higher student engagement when compared with lecture [12] and may help students develop a sense of community and lead to better performance in later classes [6]. This community results in a more inclusive learning environment and a feeling of belonging. Recent longitudinal results show that the flipped lightweight teams approach leads to fewer female and minority first-year students changing out of the computing major [13].

In this paper, we present our course evolution towards a more inclusive and successful experience for students, through a flipped, highly structured, team-based course design.

II. Class Description

The class focus is on programming data structures, with minor emphasis given to algorithms and analysis. Topics covered include: Collections, Abstract Data Types, Generics; Unit Testing; Big O Analysis; Comparing, Sorting & Searching; Stacks & Queues; Lists; Recursion; Trees & Heaps; Graphs; Hash Functions; and Algorithm Strategies. The lead instructor also engaged in periodic meta-education, explaining why the class was structured as a flipped, active learning classroom, referring to learning sciences literature on constructivist learning, peer instruction benefits, and quizzing as an effective learning tool. Figure I shows an overview of course details over four semesters, and the top row shows the short form semester names.

978-1-5386-1174-6/18/$31.00 ©2018 IEEE

TABLE I
EVOLUTION OF COURSE FEATURES ACROSS THE FOURS SEMESTERS OF THE FULLY FLIPPED, ACTIVE LEARNING VERSION OF DATA STRUCTURES, INCLUDING CLASSROOM FEATURES, STUDENT TEAM SIZES & FORMATION, THE TEACHING TEAM, LEARNING RESOURCES, CLASS ACTIVITIES AND DATA COLLECTION. SEQUENTIALLY COLORED AND UNCOLORED ROWS ARE RELATED.

	Fall 2016 (F16)	**Spring 2017 (S17)**	**Fall 2017 (F17)**	**Spring 2018 (S18)**
Enrollment at census/end	88/77	51/49	124/115	74/68
Classroom	Large active learning class. 14 large round tables seating 9 people each.	Medium active learning class. 14 small round tables seating 4-5 people each.	Large active learning class. 14 large round tables seating 9 people each.	Medium active learning class. 75 individual rolling chairs with table & under chair storage.
Classroom Tech.	3 laptops/table. HDMI screen, whiteboard at each table. 5 projection screens around the classroom.	No laptops provided. 4 projection screens (one per wall). 2 whiteboard walls.	3 laptops/table. HDMI screen & whiteboard at each table. 5 projection screens around classroom.	No laptops provided. 5 projection screens with interactive whiteboards spread through classroom
Teaching Team	2 professors (1F, 1M), 2 grad students (1F, 1M)	1 professor (F), 1 grad student (M), 1 undergrad (M)	1 professor (F), 2 grad students (2F), 3 undergrads (1F, 2M)	1 professor (F), 1 grad student (F), 3 undergrads (1F, 2M)
Team sizes	6-8	4-5	4-5	5
Team formation	Random, females in pairs	Random, females in pairs	Random, females in pairs, minorities in pairs	Random, females in pairs, minorities in pairs, attempt to balance by skill
Textbook	Lewis & Chase "Java Software Structures"	Lewis & Chase "Java Software Structures"	zyBooks interactive textbook, combines intro CS and DS	zyBooks interactive textbook, combines intro CS and DS
Discussion Forum	Canvas Discussion Forum	Canvas Discussion Forum	Piazza, allowing anonymous posting	Piazza, allowing anonymous posting
Sketchbooks	Required, use encouraged, not graded	Required, use encouraged, not graded	Required, graded	Required, graded
Peer Instruction Quizzes	Turning Technologies Clickers	Turning Technologies Clickers	Poll Everywhere, with word cloud warm-up questions	Poll Everywhere, with word cloud warm-up questions
Programming Labs	In class pair-programming	In class pair-programming	In class pair-programming	In class pair-programming
Data Collection (including grades)	Mid-semester and late-semester anonymous feedback surveys	Background Survey, Mid-semester and late-semester anonymous feedback survey	Background Survey, Learning reflections, Late-semester anonymous feedback survey	Background Survey, Learning reflections, Late-semester anonymous feedback survey

A. Teaching Team and Stepping Stone Mentorship

During the first offering of this flipped course in F16, the class was co-taught by two professors and two TAs, see Figure I. The dual teaching team for the first semester has been a common approach in our college when fully flipping a course for the first time. There are many moving parts, and many new components to be developed. Having two professors collaborate on this splits the workload and exposes more faculty to the materials and approach.

Active learning classrooms require instructional guides to be moving about the room, supporting students as they work with the course content. This requires teaching assistants (TAs) trained in the active learning methodology and excited to work with students. Teaching continuity can be challenging to maintain across semesters when TAs graduate or change assignments. For this reason, TA mentorship across semesters has become integral to our course success. During the later three semesters, the class was taught by one female professor (lead author) with the assistance of a combination of graduate and undergraduate TAs. The undergrad TAs were recruited from students taking the class in previous semesters and were mentored by one of the graduate TAs (co-author), which aligns with the stepping stone approach to mentoring [14].

B. Prep-work

The course is a flipped class meaning that there are no lectures, with students doing active learning activities during class time. Students consume content information at home through prep-work. Each week, students were required to read a section of the textbook (and in recent semesters complete the interactive activities), watch some videos, and then take an online quiz on Canvas. The graded quiz was due before class, but only available after students completed the textbook assignments and visited the video pages.

C. Materials & Auto-grading

We assigned Lewis & Chase's 'Java Software Structures' [15] in the first two semesters and then switched to a zyBook interactive online textbook in the recent two semesters. The zyBook combines an introductory Java programming book[1] and a data structures book[2]. This combined text allowed the instructor and TAs to direct students struggling with programming to chapters and interactive exercises that could help students master those earlier concepts that they were struggling with. Students were also required to have a sketchbook to bring to class for use in drawing diagrams of data

[1]"Java Early Objects" by Adrian Lizarraga and Roman Lysecky: https://www.zybooks.com/catalog/java-early-objects/

[2]"Data Structures Essentials" by Roman Lysecky and Frank Vahid: https://www.zybooks.com/catalog/data-structures-essentials/

structures. The course was managed through the Canvas LMS, which provided weekly structure for the prepwork materials, including links to videos, as well as lab and assignment materials. Students used the NetBeans IDE and also made use of web resources, including CodingBat[3] and visualgo.net[4] for some active learning activities.

Students were required to use the Web-Cat system [16] for auto-grading of programming assignments and some in-class labs. Web-Cat offers many benefits: instead of waiting for help sessions and lab-time, students get immediate, real-time help with their assignments via Web-Cat's feedback system and our pre-written hints. This allows students to improve their code quickly and iteratively, acclimating students to the real-world experience of coding, self-testing and then committing their code to a system which will run test cases against it. Automated grading also freed up our teaching staff to provide one-on-one support to students who were struggling. Web-Cat provides a grading system based on test cases, which means that students from under-represented groups cannot be unconsciously discriminated against by teaching staff during grading.

D. Classroom Teams

Inclusive, intersectional design advocates negotiating differences rather than downplaying or dismissing them [17] and the 'contact hypothesis' suggests that prejudice between different groups can be mitigated through contact in the right conditions [18]. Teaching students how to navigate differences is crucial in developing critical consciousness for others and for knowledge. We made use of Lightweight Teams [6], with teams of students created randomly, to attempt to create an atmosphere that allows for such consciousness to develop. The use of lightweight teams has many benefits, including extended communication practice, peer learning, and the development of stronger social ties with other students. We tweaked the teams slightly in an attempt to ensure that there were no teams with a lone female student. Teams were assigned before the first class and stayed together at an assigned table throughout the semester. In F17 we also began to attempt to ensure that our under-represented minority students were not alone on a team, and to put them on teams in pairs or threes. In the current semester, we have begun to try to also balance teams by programming skill, to ensure that less experienced programmers are on teams with more experienced programmers. Beginning in F17, we also began to ask students for their preferred pronouns in the background survey completed on the first day of class.

E. Communications

In the first two semesters, we made use of the Canvas Discussion Forum. This was in line with our communications policy, in which students were instructed to post all course-content-related questions to the forum. This provided other students the opportunity to help answer questions to further encourage peer instruction. Additionally, by asking questions on the discussion forum, all students were given equal access to the answers, ensuring information transparency. The instructional team only responded to email about issues pertaining to an individual student, such as absences or grades. In the two most recent semesters, we transitioned to using the Piazza discussion forum, which allows grouping of students by teams and anonymous posting. The anonymous posting can be helpful for students who lack confidence or feel embarrassed to ask questions.

F. Classrooms & Technology

In the F16 and F17 semesters, the class was taught in a large active learning classroom that has 14 large round tables (each seating 9 people and providing three laptops). Each table had its own whiteboard or glass wall to write on and a large LCD screen. This screen could display the projection from the professor, or display from a laptop, allowing group work around a shared screen. There were also large projection screens located around the room, ensuring that students sitting at any angle could see what the professor was projecting. This classroom was large enough that students at one end of the room could not easily see or hear students at the other end, so each table had a microphone to enable class-wide discussion.

In the S17/18 semesters, the class was taught in smaller, retro-fitted active learning rooms. These rooms have screens on all walls and a variety of whiteboard surfaces, but no laptops. In S17, the students sat at small round tables, which worked reasonably well. In S18 students sat at mobile desks that could be arranged into groups. This worked less well as students would not necessarily stay close together in groups, and the classroom was difficult to navigate.

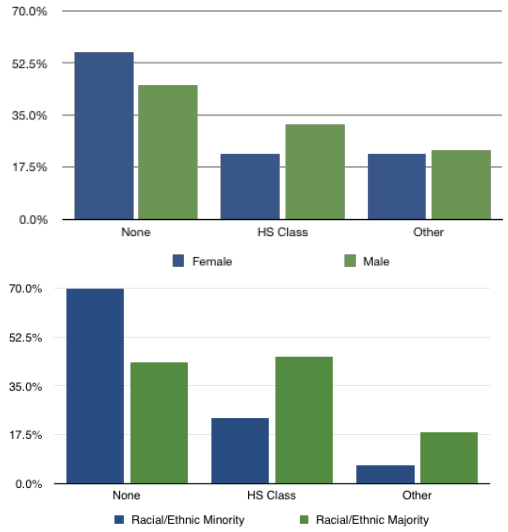

Fig. 1. The percentage of students reporting experience in programming prior to entering college, split by gender (top) and racial/ethnic minority vs non-minority (bottom), averaged across semesters. Minority students in this case are African-American or Latinx.

[3]codingBat.com
[4]visualgo.net

III. Results

In this section, we present results from non-anonymous student background surveys, anonymous student feedback surveys, as well as aggregated student performance data.

A. Background Experiences

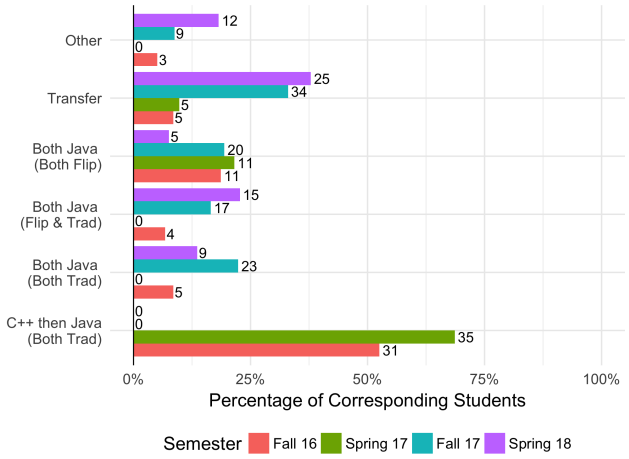

Fig. 2. Students categorized by their self-reported background courses, by semester, demonstrating the wide variety of prior course experiences.

In the F16 semester, we observed that numerous students struggled and told us about very diverse levels of prior programming experience. Thus, at the beginning of the S17, F17 and S18 semesters, we asked students to complete a non-anonymous background survey on the first day of class to gain an understanding of the diversity of their prior computing experiences. This was a mandatory in-class activity. Results demonstrate that there is significant variation in the background experience of students with respect to whether or not they learned any programming *before* college. Figure 1 shows the variation in programming experience prior to college, split out by gender (top), and split out by minority race/ethnicity vs non-minority status (bottom). In this paper, we define our minority population as either African-American or Latinx students. Female students are less likely to have taken a formal high school computing course than males. The difference for race/ethnicity is more pronounced with 70% of racial/ethnic minority students having no exposure to programming prior to coming to our program. Our consists almost entirely of students majoring in our college with as many as 1-2 non-majors each semester.

The first two programming classes are a pre-requisite for students taking the data structures class, but at our institution first-year students' experiences in these courses can vary substantially by section. Some students took both of their first programming courses in a fully flipped, team-based environment similar to the pedagogical model for our course. Those students all learned object-oriented Java in both the earlier courses. Some students took both their first programming courses in traditional lecture sections, learning procedural C++ and then Java. Some students experience a combination of those. In addition to students who have completed their first two programming courses at our institution, there are also a significant portion of transfer students who have taken programming courses as part of an associate's degree at one of several regional community colleges. Thus, there is a huge variation across our student population in terms of how their experience in the first two courses may have prepared them for the data structures course. Figure 2 shows the breakdown of prior courses taken by students across the last three semesters.

B. Survey Feedback

We collected anonymous mid-semester (F16 and S17) and late-semester feedback (F16, S17, F17, S18) through Canvas. There was no compensation for completing these surveys and response rates varied, depending on whether students were given time in class to respond. For F16, mid-semester response rate was 82%, end of semester response rate was 41%. For S17, mid-semester response rate was 16% and end of semester response rate was 37%. In F17, we stopped doing the mid-semester survey as we introduced learning reflection exercises, but the end of semester response rate was 90%. The S18 end-of-semester survey response rate was 93%.

TABLE II
Preparedness Perceptions, categorized by Prior Course Experiences as surveyed in Fall 2016.

Previous Courses	**Perceived Preparedness** (5=Strongly Agree)
Transfer student (n=5)	3.4
1 course in C++ (traditional), then 1 course in Java (traditional) (n=31)	3.4
1 course in C++ (traditional), then 1 in Java (flipped) (n=3)	2.3
1 course in Java (flipped), then 1 course in Java (traditional) (n=4)	3.8
2 courses in Java, both flipped (n=11)	4.2
2 courses in Java, both traditional (n=5)	2.8
Other (n=3)	3.6

We asked students to report about their prior two programming class experiences and to rate their agreement with the following statement: "I feel like the courses I took previously have given me adequate preparation for this course." Responses were on a 5-point Likert scale with 5 meaning 'strongly agree', 3 meaning 'neutral', and 1 meaning 'strongly disagree'. Table II shows the results, averaged across the F16 and S17 semesters, when we did mid-semester feedback surveys. Students who felt the most prepared were those who had previously taken two fully flipped courses in Java. The students who felt the least prepared were the students who had taken a traditional C++ course followed by a flipped Java course and students who took two Java courses, but both as traditional lecture/lab courses.

1) Class experience: We asked students to rate their agreement that their learning was enhanced by particular course components, including the use of Web-Cat, the use of sketching and the assigned prep-work. Responses, as shown in

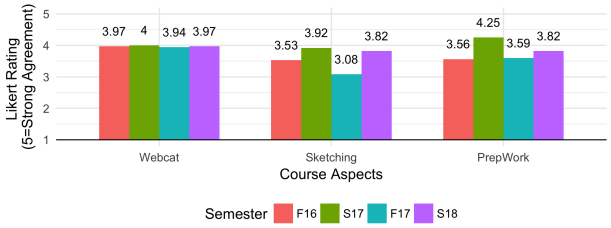

Fig. 3. Students' averaged agreement ratings about the effectiveness of various aspects of the course, using a 5 point Likert scale, with 5 meaning 'strongly agree' and 1 meaning 'strongly disagree'.

Figure 3, were on a 5 point Likert scale with 5 meaning 'strongly agree'.

2) Team Experience: We asked students about their experience with respect to lightweight teams. In particular, we asked their agreement with the statement "I like being part of a team and sitting with my team each week." Figure 4 shows the number of students, across semesters, who chose each level of agreement for this statement.

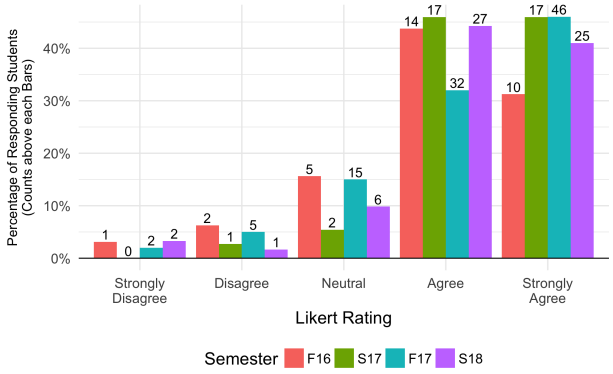

Fig. 4. The number of students choosing each option in response to the statement about liking being on a team, across all four semesters.

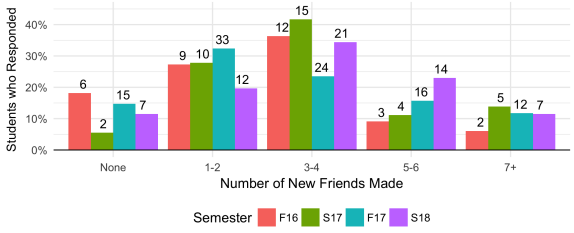

Fig. 5. The percentage of survey respondents who reported making various numbers of new friends, split out by semester. For example, 18% of F16 respondents made no new friends, 27% reported making 1 or 2 new friends.

We also asked students how many new friends they made as a result of taking this class. Figure 5 shows the percentage of students who reported making specific numbers of new friends, averaged across the four semesters.

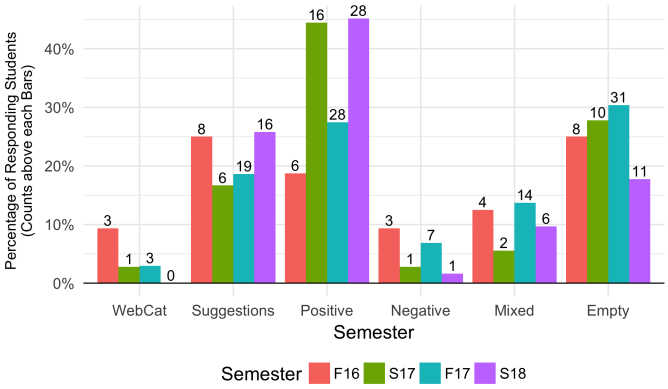

Fig. 6. Counts of different types of freeform comments in the anonymous feedback surveys across semesters.

3) Open-ended Responses: Students could also provide open-ended feedback in the surveys. Some students left this empty, but others provided positive, negative or mixed criticism, or suggestions or Web-Cat complaints, as shown in Figure 6. The comments were categorized by the authors. Positive praise increased over the four semesters and the percentage of purely negative criticism was quite low overall. Examples of positive praise include:

> It was a great experience overall. The flipped class system is remarkably efficient in my humble opinion. Thanks for this semester.
>
> I liked the way the class was conducted and felt that it was a very welcoming learning environment where there wasn't any judgement because of how everyone in the class had different backgrounds in programming and how you were always open to questions during lectures which was very helpful.

Examples of negative critique include:

> I know teachers really like the flipped classroom, but I really don't like data structures as a flipped classroom. I think the teacher is very nice and this is not a reflection on them, I just do not like flipped classrooms I do not believe that a class as important as data structures should be flipped for me.
>
> Overall, I think the flipped structure hurt me more than it helped.

Examples of mixed feedback include:

> I feel like occasional lectures would help to reinforce information retention over the duration of the class. I feel the individual assignments were slightly too challenging... Overall this was my favorite class and the way it was conducted is far superior to any other class I have taken.
>
> I really found this class interesting! I can't think of anything to improve on, except the lab activities took too long to complete.

TABLE III

PERFORMANCE DATA. DFW RATE IS THE PERCENTAGE OF STUDENTS RECEIVING A 'D', AN 'F' OR WITHDRAWING FROM THE CLASS AFTER THE DROP/ADD DEADLINE. IN THIS ANALYSIS, MINORITIES ARE AFRICAN-AMERICAN OR LATINX STUDENTS AND NON-MINORITIES ARE STUDENTS NOT FITTING THAT CRITERIA. DIFFERENCES IN STUDENT NUMBERS BETWEEN THE CENSUS DATE AND THE END OF THE SEMESTER REPRESENT STUDENTS WHO WITHDREW FROM THE COURSE.

Category	Group	Fall 2016 (F16)	Spring 2017 (S17)	Fall 2017 (F17)	Spring 2018 (S18)
Start/End	Total Enrollment	88/77	51/49	124/115	74/68
	Females	11/10 (13%/13%)	6/6 (12%/12%)	12/12 (10%/10%)	9/7 (12%/10%)
	Minorities	20/18 (23%/23%)	12/12 (24%/22%)	17/16 (14%/14%)	12/10
	Female Minorities	4/4 (5%/5%)	1/1 (2%/2%)	1/1 (1%/1%)	2/1 (1%/1%)
DFW Rate	Overall	27%	24%	13%	(estimate: 17%)
	Female	9%	17%	0%	(estimate: 16%)
	Minorities	25%	50%	18%	(estimate: 17%)
	Female Minorities	0%	100%	0%	50%

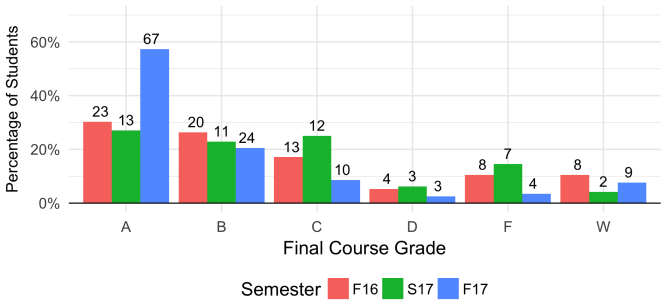

Fig. 7. This barchart shows the percentage of students getting As, Bs, Cs, Ds, Fs and the number of withdrawals over the two semesters.

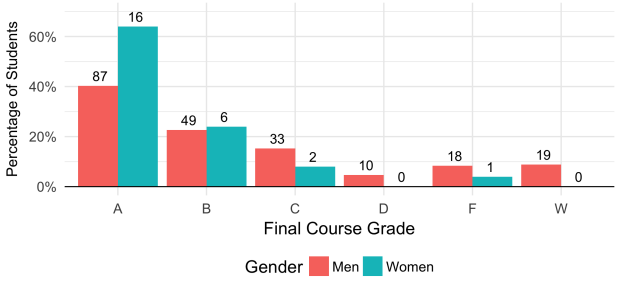

Fig. 8. The percentage and number of students receiving As, Bs, Cs, Ds, and Fs, combined over three semesters (F16, S17, F17), split by gender.

C. Student Performance

At the time of this writing S18 students had not written their final test and thus final grades were not available, thus we were only able to estimate DFW rates for that semester, and our performance analysis focuses on the first three semesters. Students across the first three semesters performed well, though the first semester had a high percentage of withdrawals, see Figure III. Female students did well in the flipped class, with very few failing or withdrawing, as shown in Figure 8. Students in the minority performed slightly worse than non-underrepresented groups, as shown in Figure 9.

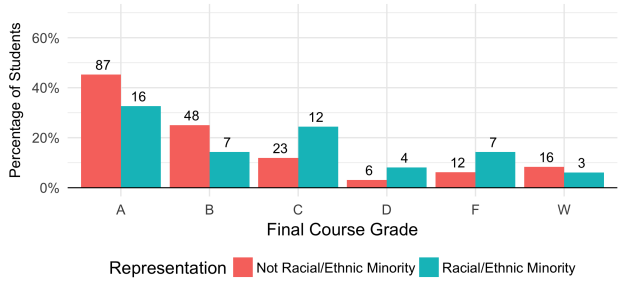

Fig. 9. The percentage and number of students receiving As, Bs, Cs, Ds, and Fs, over three semesters (F16, S17, F17), split by racial/ethnic status.

IV. DISCUSSION

Here we discuss our interpretations of the performance data. Our own classroom observations help to provide a more nuanced understanding of what is happening with respect to inclusivity. We argue, based on survey results and observations, that there are eight main components that impacted classroom inclusivity.

The average DFW rates for this class in our college have historically sat at around 25%, but have sometimes been over 50%. While the DFW rate in the first semester was above that average, it was also the first time the class had been taught at that scale and in this way, so there were some issues that had to be ironed out. We were happy to see the DFW rates drop the following term, and to really drop by Fall 2017 down to 13%. We attribute these improvements to major changes that improved student outcomes, including the use of the interactive textbook, required sketching activities in class, the move to the Piazza forum, the cultivation of teaching team, as well as further refinements in lab activities.

We are most interested in the performance of female and ethnic/racial minority students who are our under-represented groups, as well as students at the intersectional location of being female *and* ethnic/racial minority. Table III breaks this down, and shows that women are very unlikely to have DFWs in this class. In Spring 2017 however, the one minority female student in the class dropped out of university, causing a

100% DFW rate at that intersectional location. The distribution of grades shows that women are doing well in our class, getting more As and Bs, and very few DFWs, Figure 8. This shows that our class structure and environment is a good learning environment for our female students. The picture for minority students isn't quite as positive, with minority students receiving fewer As and Bs and more Cs, Ds and Fs than non-minority students, Figure 9. Our efforts to help racial and ethnic minority students succeed in this class are ongoing.

A. Reducing Human Bias in Grading

Using the Web-Cat system for auto-grading class assignments caused significant headaches when it was first adopted, but its use was tweaked in subsequent semesters. Students were typically given 30 submissions for each assignment and while students with significant prior programming experience might only use 5 or 10 submissions, students with much less experience could use all the submissions available to iterate and improve their code. Most students seemed to score better on programming assignments and require less face-to-face additional help because of the iterative nature of the multiple submission system. Assignments were worth 40% of students' final grades, so it was important that this grading was less biased by human judgement.

In addition to the Web-Cat system, students had four tests throughout the semester, which were mainly multiple choice tests of understanding and application (but not code writing). These tests were also auto-graded by the LMS. Students who had perfect attendance in the class could also opt to retake their lowest test during the final exam period, providing them with an opportunity to revisit material they had struggled with earlier in the semester and improve their test grade.

In other attempts to reduce human bias, we provided multiple choice prep-work quizzes on Canvas and use zyBooks interactive textbook activities. Sketchbooks were graded by a TA, but using a well-defined rubric. Labs were graded by TAs and the professors in class. The labs and sketchbooks were thus the only part of student grades subject to human bias, and these were worth only 15% of the students' final grades.

B. Teams & Social Learning

We have taught flipped classes with lightweight teams with many team sizes now, and this experience confirms that while some teams of 4 can work, 4 is often too small and 7+ is definitely too big. Because the teams are lightweight, with no associated grades, students get the benefits of peer instruction and social learning, but without the stress associated with high-stakes team projects. Students appear to appreciate the lightweight teams structure, as shown by positive ratings about being on a team, see Figure 4. We also see that students are making new friends, which should help them feel more like they belong in the class, see Figure 5. Students who report making more than four friends are typically making friends because of the interaction that goes beyond their team during peer instruction quizzes. We continually observe some teams extending their interactions outside of class, working on prep-work and studying for tests, which likely enhances learning.

Because the teams are formed randomly, we expect that many students in the class are interacting with students who are not like them. By creating diverse teams and facilitating cross-team interaction, we encourage students to negotiate differences in knowledge and identity. Additionally, talking and listening to other people who have different points of view allows students to engage in critical analysis of knowledge on a personal level. The 'contact hypothesis' suggests that the high level of interaction in our teams should act to mitigate discomfort majority group students might have with people who are 'other' [18]. Unfortunately, we still have a number of teams that are all male, or with no minority students, simply because there is still not a critical mass of women and minority students in our major.

C. Active Learning Classrooms

Physical teaching space had significant impacts on the course. The classroom size, and especially the table size, impacted the size of the teams, and through that, the quality of the learning experience. In order to promote the most social learning experience, we found that classrooms with round tables for students to sit at together work best. We also found it important for there to be enough space around the tables for students to get up and walk around. The S18 classroom, with rolling desks, was very crowded and that limited social interaction *between* teams, meaning that students were mostly only interacting with others on their own team.

D. Intense Structure Linked to Grades

There are many activities in the course that contribute to the students' final grades. This is unlike a traditional course with just a couple of assignments, a midterm and final. In our course, participation is required, attendance is taken, points are given for completing the prepwork quizzes, sketchbooks of data structures, multiple non-cumulative tests, and the interactive textbook activities. Practice programming labs happen during class hours and also count towards the final grade. Thus, the level of structure – of mandatory, scheduled, sequenced activities – is very high. This level of structure forces students to do a lot of work, and by doing a lot of work, students likely learn more. This adds to the evidence found by Eddy and Hogan, who have shown that high levels of course structure help students who are under-represented and/or first generation college students [7]. The intense structure may annoy some students, but it certainly does not appear to hurt any students, and most students adapt quite well to it.

We observed that some students struggled with the more involved programming assignments and labs, especially those who came in with less experience. In the prior two CS classes there are 3 hours of class time, *plus* three hours of lab time. The data structures class has only 3 hours of class time, and no separate lab time. This means students are doing pair programming assignments during part of the 3 hours of class, and that may not provide enough practice. As Luxton-Reilly

pointed out, what many students need to succeed in computing is simply more time, practice and exposure [19]. The prep-work forcing functions, in addition to providing structure, also simply force students to spend more time working with the data structures content. While this helps students who have less prior experience from high school, it may also put a strain on students who work part-time or full-time jobs to pay for their education and support themselves.

E. Intentionally Supportive Culture

We have intentionally strived to create a casual, fun, social and supportive culture inside and outside of the class. Each class begins with an anonymous warm-up question (such as 'What's your favorite side dish at Thanksgiving?'), with a projected word cloud of responses. These questions are aimed at getting students to be social and talk to one another. The professor is not afraid to be a bit silly, to make mistakes in front of the students, and welcomes challenges and corrections. The TAs also interact in class and have fun with the students. The professor asks students frequently about how they are doing and empathizes with their stress levels and time demands. This friendly atmosphere is meant to relax students so that their anxiety is reduced and they can learn more easily [20].

F. Engaged, Mentored TAs

The teaching team truly acts as a team, with the TAs taking on increasing responsibility for assignment design and testing, interacting heavily with students during class and offering frequent, collaborative help sessions instead of traditional office hours. Students who came to help sessions spent time working together, helping each other, and building relationships. In response to a reflection question about what was most surprising about this class, one student responded: "The effectiveness of the teaching assistants... I really had no idea that a class could actually be accentuated by TAs." The TAs mentored one another across semesters, were active in the classroom and during help sessions, and helped to create a positive learning environment. The undergrad TAs typically have more time to help students, and the smaller gap between age and progress through the major likely increases approachability. The first undergrad TA (co-author) has heard from students expressing interest in research and TA positions, and many have found TA positions in the department.

G. Transparent Communications

We have a strict communications policy, requiring all course questions to be asked on the public discussion forum. This information transparency ensures that students who have the confidence to ask a lot of questions are not privy to extra information, rather, all students benefit from the answers. In F16 and S17, discussions were held on Canvas, but that discussion forum is slow and unintuitive, and it does not allow anonymous posting. The move to the Piazza discussion forum allowed students to post questions anonymously and many students did. We believe this helps students feel more comfortable asking questions, because they do not have to worry about looking dumb in front of their classmates.

H. Low-Cost Resources

Economic disadvantage can play a role in decreasing inclusive success. As much as possible, we have tried to ensure that required resources are free, or low-cost. The required zyBook subscription is approximately $78 for the combined book. This is the official adopted textbook and so students can use financial aid to pay for it. Other resources used (Netbeans, visualgo.net, Web-cat, etc.), are free. The other major resource issue is laptop access. The majority of our students have their own laptops, but some students may only have a desktop computer. In pair programming, we can always ensure a student who does not have a laptop is paired with someone who does. But students who have to get updated files from their pair programming partner and put them in the right place on their home desktops sometimes struggle. If they have file system or project setup issues, they are not able to bring their system in to a TA for assistance.

I. Limitations

Education has the potential to both oppress and liberate [17]. Paulo Freire created some of the first work on this idea, examining how education both disenfranchised and empowered students in various regions of Brazil [21]. Freire's work predated intersectional inquiry by approximately twenty years but focused on intersecting inequities of class, race, ethnicity, and citizenship. We have structured this course with these ideas in mind, but are unable to do much intersectional analysis because of the lack of critical mass or the inability to gather sensitive data pertaining to students' identities. We have not yet considered other intersectional identity markers in our analysis (e.g., ability, age, religion, sexual orientation, socio-economic class). Nevertheless, we are committed to creating a course that encourages inclusive success.

V. Summary & Future Work

We have contributed a detailed description of the evolution of a data structures course over four semesters, with the goal of creating a more inclusive, successful learning experience for all students. We have presented student background data, performance data and anonymous feedback to tell the story of an inclusive classroom. We argued that there are eight main components that can positively impact inclusivity in the data structures classroom: reduced human-bias in grading, teams & social learning, active learning classroom space, intense structure linked to grades, intentionally supportive culture, engaged and mentored TAs, transparent communications, and the use of low-cost resources.

In the future, we hope to be able to perform more intersectional analyses, as the number of women and minorities grow to a critical mass. We also want to examine how this structure impacts students who are first generation college students or from low socio-economic status backgrounds. We also have plans to implement some adaptive learning modules early in the class to address the wide disparities in preparation.

REFERENCES

[1] Alvaro E. Monge, Cameron L. Fadjo, Beth A. Quinn, and Lecia J. Barker. EngageCSEdu: Engaging and retaining CS1 and CS2 students. *ACM Inroads*, 6(1):6–11, February 2015.

[2] James Kurose. Booming undergraduate enrollments: A wave or a sea change? *ACM Inroads*, 6(4):105–106, November 2015.

[3] Lecia Jane Barker, Kathy Garvin-Doxas, and Michele Jackson. Defensive climate in the computer science classroom. *ACM SIGCSE Bulletin*, 34(1):43–47, 2002.

[4] Colleen M Lewis, Ruth E Anderson, and Ken Yasuhara. I don't code all day: Fitting in computer science when the stereotypes don't fit. In *Proceedings of the 2016 ACM Conference on International Computing Education Research*, pages 23–32. ACM, 2016.

[5] Regina Deil-Amen. Socio-academic integrative moments: Rethinking academic and social integration among two-year college students in career-related programs. *The Journal of Higher Education*, 82(1):54–91, 2011.

[6] Celine Latulipe, N Bruce Long, and Carlos E Seminario. Structuring flipped classes with lightweight teams and gamification. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, pages 392–397. ACM, 2015.

[7] Sarah L Eddy and Kelly A Hogan. Getting under the hood: how and for whom does increasing course structure work? *CBE-Life Sciences Education*, 13(3):453–468, 2014.

[8] Scott Freeman, Sarah L Eddy, Miles McDonough, Michelle K Smith, Nnadozie Okoroafor, Hannah Jordt, and Mary Pat Wenderoth. Active learning increases student performance in science, engineering, and mathematics. *Proceedings of the National Academy of Sciences*, 111(23):8410–8415, 2014.

[9] Aditi Kothiyal, Sahana Murthy, and Sridhar Iyer. Think-pair-share in a large CS1 class: Does learning really happen? In *Proceedings of the 2014 Conference on Innovation & Technology in Computer Science Education*, ITiCSE '14, pages 51–56, New York, NY, USA, 2014. ACM.

[10] Cheryl A Dugas. No computers? No problem! Active and cooperative learning in an introductory computer science course. In *Frontiers in Education Conference, 2008. FIE 2008.*, pages 3–16. IEEE, 2008.

[11] Graciela Gonzalez. A systematic approach to active and cooperative learning in CS1 and its effects on CS2. In *Proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '06, pages 133–137, New York, NY, USA, 2006. ACM.

[12] Stephen MacNeil, Celine Latulipe, Bruce Long, and Aman Yadav. Exploring lightweight teams in a distributed learning environment. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*, SIGCSE '16, pages 193–198, New York, NY, USA, 2016. ACM.

[13] Celine Latulipe, Audrey Rorrer, and Bruce Long. Longitudinal data on flipped class effects on performance in CS1 and retention after CS1. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, pages 411–416. ACM, 2018.

[14] Eric S Roberts, Marina Kassianidou, and Lilly Irani. Encouraging women in computer science. *ACM SIGCSE Bulletin*, 34(2):84–88, 2002.

[15] John Lewis and Joseph Chase. *Java Software Structures: Designing and Using Data Structures*. Addison-Wesley Publishing Company, 4th edition, 2014.

[16] Stephen H Edwards and Manuel A Perez-Quinones. Web-cat: automatically grading programming assignments. In *ACM SIGCSE Bulletin*, volume 40, pages 328–328. ACM, 2008.

[17] Patricia Hill Collins and Sirma Bilge. *Intersectionality*. John Wiley & Sons, 2016.

[18] John F Dovidio, Peter Glick, and Laurie A Rudman. *On the nature of prejudice: Fifty years after Allport*. John Wiley & Sons, 2008. article about contact hypothesis workd.

[19] Andrew Luxton-Reilly. Learning to program is easy. In *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education*, pages 284–289. ACM, 2016.

[20] Joseph A Durlak, Roger P Weissberg, Allison B Dymnicki, Rebecca D Taylor, and Kriston B Schellinger. The impact of enhancing students social and emotional learning: A meta-analysis of school-based universal interventions. *Child development*, 82(1):405–432, 2011.

[21] Pablo Freire. *Pedagogy Of The Opressed*. Bloomsbury Academic, 1970.