Moving from Managing Enrollment to Predicting Student Success

Deborah A. Trytten
School of Computer Science
University of Oklahoma
Norman, OK USA
dtrytten@ou.edu

Amy McGovern
School of Computer Science
University of Oklahoma
Norman, OK USA
amcgovern@ou.edu

***Abstract*—We describe the construction and assessment of a plan to foster student success in Computer Science (CS) in response to continued enrollment growth. We examined cross correlations of grades from student transcripts from the past four years to determine what patterns of grades in early classes were indicative of future success. The resulting statistics and visualizations showed that students generally did better in the first class than subsequent classes, suggesting that C grades in the first class might indicate difficulty. We also examined students enrolled in the pre-capstone software engineering course. The analysis showed that no graduating CS major had received a C in both first two programming courses. We examined the data to be sure that the proposal would not diminish the participation of members of underrepresented groups disproportionately. As a result, the CS faculty approved a change in prerequisite for the third course to require that student receive an A or a B in at least one of the first two courses. GPA data was also analyzed and found to be a poor predictor of success in computer science, demonstrating that our previous approach to limiting enrollment was inferior. This new approach has advantages over the previous GPA-based plan, but enrollment management plans risk making college harder for at risk students to succeed in CS. The biggest limitation of this plan is that it is expected to make only a small change in CS enrollment.**

Keywords—enrollment management; retention; Computer Science;

I. INTRODUCTION AND RELATED WORK

Computer science (CS) enrollment historically has surged and faded periodically depending on the health of the technology sector and availability of high paying jobs. We are currently in a nationwide enrollment boom [1]. Whether or not this enrollment boom is permanent or part of yet another cycle remains to be seen, although many in the CS education community believe that this enrollment surge will continue because the pervasiveness of computing is now obvious to college age students [1].

As part of this enrollment surge, many CS departments are now seeing record growth in the number of majors, minors and non-majors taking CS classes. Increasing enrollment presents substantial pressures on CS departments. Few institutions have been able to increase faculty size proportionately with the enrollment increases [1]. Hiring CS faculty is particularly challenging as institutions must compete with high paying and exciting industry opportunities. Hiring is also especially daunting in high need areas like cybersecurity [2]. The Taulbee survey reports that, in 2015, a near-record high of 57.3% of doctoral graduates took positions in industry [3]. In addition, many public institutions of higher education, including ours, are facing austere budgets. Mounting uncertainty about the future of research funding in the United States also creates university-level budget anxiety [4]. Budget crises make administrators wary of responding to what could be temporary enrollment growth with permanent positions, especially on the tenure track [5].

A recent survey was conducted of both public and private institutions of higher education to see how departments are responding to enrollment growth in CS. Policies that are in current use include (in order of popularity): significantly increasing class sizes, increasing the number of sections offered, increasing summer offerings, reducing the number of low-enrollment courses offered, reducing offerings to non-CS majors, increasing blended offerings, increasing online offerings, raising the bar on prerequisite grades, and spinning off service courses [6]. While our work would fall into the category of raising the bar on prerequisite grades, our decision to base the changes on student success data is different than simply choosing prerequisites based on enrollment number targets.

Enrollment management decisions are especially important for all CS departments because past enrollment booms in CS have not been kind to members of underrepresented groups in general, with data for women being most available [7, 8]. Some of the policies above (e.g. reducing offerings to non-CS

978-1-5090-5920-1/17/$31.00 ©2017 IEEE

majors, raising the bar on prerequisite grades) have historically decreased the availability of CS to women [7].

The history of enrollment at the University of Oklahoma's School of Computer Science mirrors the general increases in U.S. CS enrollment nationwide. During the dot.com enrollment boom, the department was able to grow the faculty modestly. During this time, the College of Engineering imposed an enrollment management plan (EMP) to keep class sizes manageable. This was done in response to anticipated resistance to a high student/faculty ratio from ABET, our accrediting agency. The EMP required students to have an overall GPA of 2.8 in order to be able to take upper division coursework in computer science. This GPA was selected to find a reasonable balance between the number of faculty and students and to maintain the existing diversity in the discipline. This plan was in place for only a few years. When the dot com bust occurred and CS enrollment decreased, the enrollment management plan was immediately rescinded.

The School did not wish to reinstate the prior EMP during the current enrollment boom because this type of enrollment management was in conflict with the current institutional desire to increase retention [9]. The timing of the GPA-based EMP was especially problematic because it only took place after students had typically completed two full years of college. Changing majors after two years of college increases the risk that courses taken will not count towards the students eventual major and that students will not graduate on time. The faculty also felt that some students who could have been successful in CS were not permitted to continue in the discipline. Because the cutoff was GPA based, we denied enrollment to some students with high grades in CS courses but with lower grades in general education or mathematics courses. We had also found that students who had a bad start in college (sometimes years earlier) were unable to raise their overall GPAs sufficiently to be allowed to continue in CS, in spite of generous grade reprieve policies mandated in Oklahoma [10]. In addition, some students were so determined to remain in CS that they strategically took one or two semesters of courses where they felt they could get easy As to meet the grade threshold, delaying the student's graduation unnecessarily.

In spite of the evidence that numerous departments are making these difficult choices, surprisingly little has been published about the decision making process. When class sizes are increased, how does a department determine how big is too big? How are departments increasing the number of sections of courses offered in times when few institutions can undertake extensive hiring (e.g. adjuncts, full time non-tenure track faculty, teaching overloads for tenured faculty)? What is the implication of these choices for student success? When prerequisite grades are increased, which course is selected and what threshold is used, and why? The answers to these questions will have an impact on which students are permitted to continue in CS, including the possibility that members of underrepresented groups will be unintentionally excluded.

One of the motivations for our work is to encourage others to publish their processes. By publishing our process, we have two goals. The first is to share useful and tested processes with others making similar decisions. The other is to encourage discussion and examination of how the various processes may contain unintended biases, especially against members of underrepresented groups. As an example, if departments are using prior programming experience as a prerequisite to filter students, this could be strongly biased against African American students [11]. If departments are increasing their filtering of incoming CS majors, it is possible that they might favor students with more extra-curricular activities in high school. These students are more likely to be middle class, thereby excluding working class and poor students who already have many challenges to achieve degrees in higher education [12]. Without having people with high levels of cultural competence examine plans, the consequences can be quite different than anticipated.

Our research methodology is based on transcript analysis. This method has been widely used, especially in the many quantitative studies done from the MIDFIELD database (e.g. [13], [14], [15]) as well as other academic repositories (e.g. [16]), and single department level analyses (e.g. [17]). Academic transcript data has also been used to support qualitative data in mixed methods studies (e.g. [18]).

II. Research Methodology

Our data includes eight semesters (four years) of transcripts for students who were in the Gallogly College of Engineering (GCOE) beginning in Fall 2012 and ending in Spring 2016. The full dataset contains 6,226 students, and includes students with majors outside of CS. The number of years of data available to study were limited by a major redesign of the institutional database completed in 2012. The data contains the grade for student's classes within the GCOE and full demographic data. The demographic data includes the self-reported gender (reported as binary in this data) and self-reported racial/ethnic group. While considerable additional information was available, the extra data was outside the scope of what was needed to create a policy for student success. Although the original data contains identifying information needed to cross-correlate students across semesters, all of the data is reported in the aggregate. The Institutional Research Board (IRB) found this research to be exempt from IRB supervision.

There are three gaps in the data. First, although students outside of GCOE take CS classes, their demographic data was not available to us. Since the focus was on ensuring success for CS majors, this missing data was considered acceptable. Second, the data did not contain Summer semesters, which led to some gaps in grades for students who took summer classes. CS has had extremely sparse offerings of summer courses, often only the beginning programming course and one course for graduate students. We categorized missing grades as 'Unknown.' Third, the data for each semester is not recorded until the third week of the semester, which means that students who enroll but drop before the third week do not appear in the

data. Although it could be fruitful to study the students who drop classes early, it was both outside the scope of our goals and unavailable to us.

For the analysis of race/ethnicity, we focused on underrepresented groups versus overrepresented groups. We defined the underrepresented groups to be any student with a self-reported race/ethnicity containing Black, Pacific Islander, American Indian, or Hispanic. The overrepresented group contained White and Asian students. If a student had multiple ethnicities listed, they were marked as underrepresented if they had any of the underrepresented groups listed. Since we studied gender separately from race/ethnicity, while women are underrepresented in engineering, they are not included in the underrepresented group unless they reported at least one underrepresented race/ethnicity.

We studied the grades of students in the introductory CS classes as well as the grades of students in the class immediately preceding the capstone class (which we will call pre-capstone), since these students are virtually certain to be graduating from our program. The grade categories that we report are A, B, C, D, F, S, AW/W, and Unknown. A, B, C, D, and F are the usual letter grades received at most institutions. S means the course was listed as satisfied, which primarily applies to students who take the advanced standing exam for the Java 1 couse. In GCOE, students must earn an A, B, or C to pass a class. At OU, students can repeat a class where they received a grade of D, W (for withdraw) or F. For consistency, we examined the final grade that they received in the class. Some students have Advanced Placement credit that enables them to skip the first programming course in CS. We also allow students to take a prior knowledge exam for a nominal fee. This exam is graded by the instructor of the course. If they pass the exam, they receive credit as if they had taken the course with a grade of S. Grades of AW/W are students who either withdrew from that course (W) or were withdrawn by the university from all courses (AW). A grade of 'Unknown' is included to cover all of the cases where the data shows a grade for the student in a later course but does not contain a grade for the earlier course. This includes students who are enrolled in the later courses during our study period but who took the earlier courses prior to this period as well as students who took a course over the summer or who transferred the credit to OU.

OU has six different grade point averages (GPA) that are used for a variety of purposes. For this work, we used the OU retention GPA. This GPA includes all courses attempted at OU. Reprieved courses, remedial courses, and PE activity courses are excluded from this GPA [19]. There are several situations that permit courses to receive a reprieve, all coming from academic forgiveness provisions passed by the Oklahoma State Regents for Higher Education [10]. The most commonly used rule is that students can request a reprieve on the first 18 credits or 4 courses that they fail, provided that the course is later successfully repeated. A second reprieve policy permits students to have one or two consecutive semesters of grades reprieved under extraordinary circumstances.

We focused our analysis on the first three courses in the CS major sequence and on the pre-capstone course. We denote the first two courses Java 1 (CS 1323 Introduction to Computer Programming) and Java 2 (CS 2334 Programming Structures and Abstractions). Together these comprise CS 1 in the CS educational literature. Java 1 teaches the fundamentals of programming including variables, loops, methods, arrays, and some object oriented programming. Java 2 goes into more depth and focuses on abstractions, inheritance, recursion, object oriented programming, and learning to write much larger programs. The third course in the sequence is Data Structures (CS 2413 Data Structures), or CS 2 in the CS educational literature. The pre-capstone course is Software Engineering (CS 4263 Software Engineering I). We examined students enrolled in the pre-capstone course in Fall 2016 instead of the capstone course (CS 4273 Software Engineering II) from Spring 2016 because the four year window of valid data available would cause more students early programming grades to be unknown. The overlap between the pre-capstone course and the capstone course in the same year is virtually 100% of the students. It is extremely rare for a student to fail either pre-capstone or capstone.

Our data contains 969 students who took Java 1, 586 students who took Java 2, and 373 students who took Data Structures. We examine joint distributions of student grades across Java 1 and Java 2 and Java 2 and Data Structures. We also have 136 students who took the pre-capstone course, 60 of whom we have grades for Java 1 and Java 2 and 78 of whom we have grades for Java 2 and Data Structures.

To analyze the transcript data, we wrote a program that read in spreadsheets containing transcript data downloaded from the university's information system. These data were processed using a Python program written by McGovern.

III. Results and Discussion

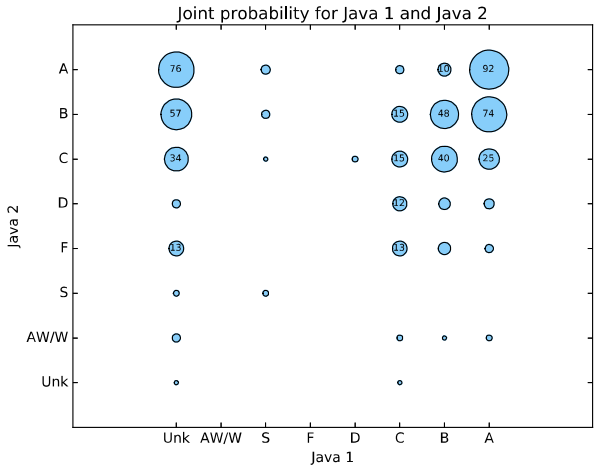

Fig. 1. Grades in Java 1 and Java 2 with the counts for each joint grade shown inside the circles. Numerical data on counts less than 10 have been omitted. The area of the circle is proportional to the count. The total count of students is 586.

Fig. 1 shows the joint distribution of grades for students who took both Java 1 and Java 2. Although we could (and did) examine grades from a single class at a time, success is easier

to determine by examining the trajectory of a student through the program. The joint distribution lets us examine how the grade in the 2nd class depends on the grade in the first class. For each of the figures, we have shown the numbers in two ways. First, if there are more than 10 students, the actual number appears inside the circle. Second, the area of the circle is directly proportional to the number of students in the joint distribution. Fig. 1 shows that an A in Java 1 primarily leads to a student earning an A or a B in Java 2 while a B in Java 1 leads primarily leads to a student earning a B or a C in Java 2. Students with a C in Java 1 are spread fairly equally across the B-F range with a small group earning an A. If a diagonal were formed from the upper right to the lower left of the square portion of Fig. 1 on the right hand side, the larger circles are found below the diagonal. This indicates that student grades tend to go down from Java 1 to Java 2. The large number of students in the Unknown category primarily represents students who took Java 1 outside our study period (the majority of the unknowns), at another institution, through advanced placement, via a prior knowledge examination, or during summer terms.

The relationship between grades in these classes was expected. Java 1's design was changed after the instructor (co-author Trytten) participated and hosted a CS Tapestry workshop [20], given by Jim and the late Joanne Cohoon. The purpose of these workshops was to collect and disseminate research proven techniques for increasing diversity in computer science. An early version of some of this work was published in 2007 [21] with a later version with more robust assessment appearing in 2011 [22]. Many of the ideas promoted in these workshops, such as having many small interactive assignments (implemented using commercial and non-commercial interactive programming tutors), and placing students without prior programming experience into separate sections of introductory courses, may tend to increase grades. For example, since the interactive tutors allow students to repeat exercises for mastery many students receive 100% on all of these exercises, substantially improving both their learning and their grades [23]. We also use pair programming on laboratory assignments, which is shown to increase both student understanding and achievement [24].

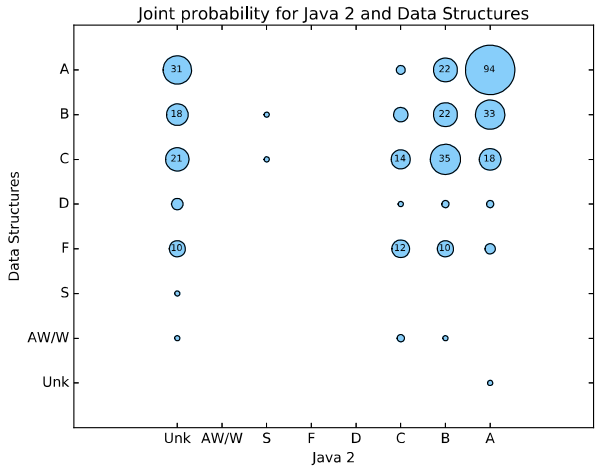

Fig. 2. Grades in Java 2 and Data Structures with the counts for each joint grade shown inside the circles. Counts less than 10 have been omitted. The area of the circle is proportional to the count. The total count of students is 373.

Fig. 2 shows the joint distribution of grades for students to took both Java 2 and Data Structures. Here the correlation between an A in the first course and an A in the second course is more clear than with Java 1 and Java 2. The majority of students who earn an A in Java 2 go on to earn an A in Data Structures and very few who earn an A will earn less than a C. Interestingly, students who earn a B in Java 2 seem to struggle in Data Structures and earn a C more often than an A or a B. Finally, students who earn Cs in Java 2 frequently earn a C or fail Data Structures. As with Fig 1., the weight of the circles below the diagonal shows that student grades tend to decrease from Java 2 to Data Structures. This is also not surprising, as difficulties with the challenge of data structures have long been known [25].

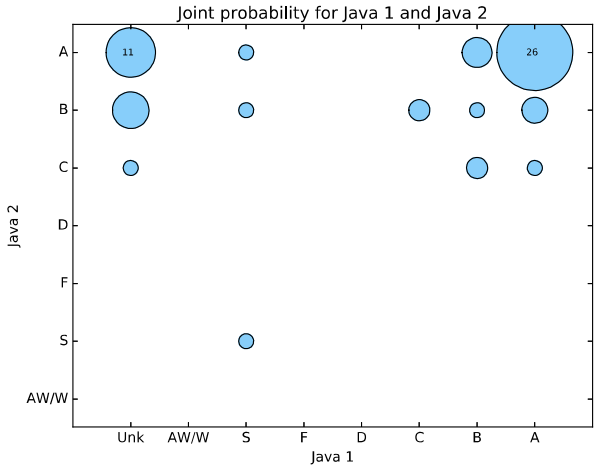

Fig. 3. Grades for students in Java 1 and Java 2, narrowed down to only students who had also taken the pre-capstone class. The total number of students represented in the graph is 60.

Fig. 3 shows the grades of the students in Java 1 and Java 2 but narrowed down to only those students who had taken the pre-capstone class. Almost all students who take the pre-capstone class graduate successfully from our program. As the graph shows, there were no students who had received a C in both Java 1 and Java 2 and had continued on to the pre-capstone class. When we presented this work to the CS faculty, we pointed out that Fig. 3 is the critical figure for making the choice to use grades in the first two classes for enrollment management.

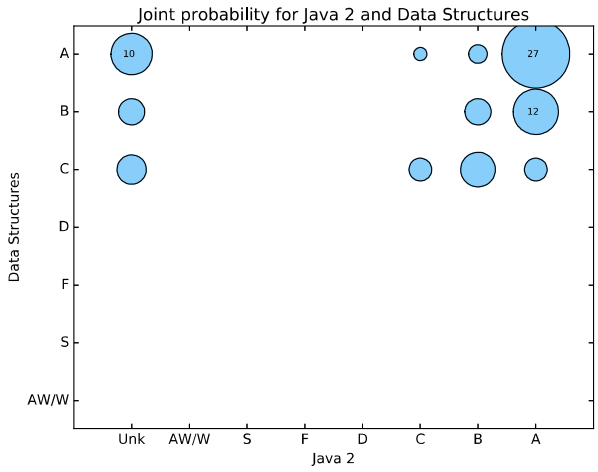

Fig. 4. Grades for students in Java 2 and Data Structures, narrowed down to only those students who had also taken the pre-capstone class. The total number of students represented in the graph is 78.

Fig. 4 shows the grades of the students in Java 2 and Data Structures, again narrowed to those students to had taken the pre-capstone class within our study period. Although there are a few students in this group who had received a grade of C in both Java 2 and Data Structures, the number is small (< 10). This indicates that the combination of Java 1 and Java 2 is a better predictor of student success than the combination of Java 2 and Data Structures.

With this data, the Undergraduate Committee solidified its recommendation to change the prerequisite to data structures to Java 2 and (a B or better in Java 1 or Java 2).

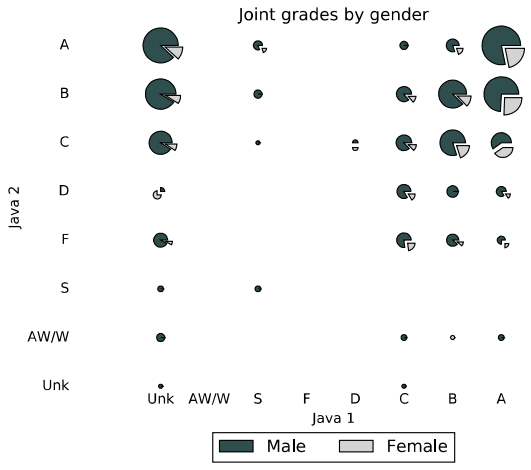

Fig. 5. Grades colored by self-reported gender for students who took both Java 1 and Java 2. The area of the pie chart is proprtional to the number of students with that joint grade.

We also wanted to ensure that the EMP approach we chose was fair to both women and members of underrepresented races/ethnicities. Fig. 5 and Fig. 6 show the same joint distribution as Fig. 1 and Fig. 2 but broken out by self-reported gender. The number of women in the CS program mirrors the national averages (75 of 457 undergraduate students are women (15.7%) [26] compared to 16.5% in national data [3]).

In each graph, the dark circle represents men and the light wedge represents women. While the sample sizes are too small to make convincing conclusions, it does appear that females who got As in Java 1 are getting disproportionately more Bs and Cs in Java 2 (Fig. 5). The number of male and female students with a C in both Java 1 and Java 2 remains small and appears to be proportionate, so our curriculum change should not disadvantage female students. Interestingly, the number of women getting As in both Java 2 and Data Structures appears to be disproportionately large (Fig. 6). These patterns could be investigated using a qualitative research methodology that is better suited to analyze small populations.

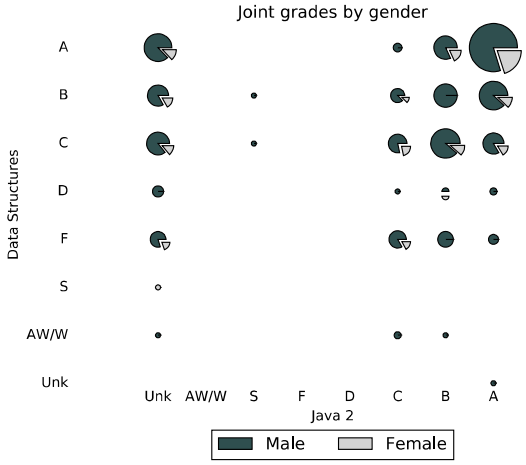

Fig. 6. Grades colored by self-reported gender for students who took both Java 2 and Data Structures. The area of the pie chart is proprtional to the number of students with that joint grade.

Fig. 7 and Fig. 8 show the grade data for Java 1/Java 2 and Java 2/Data structures respectively but broken into underrepresented and overrepresented racial/ethnic groups (women are not counted as underrepresented). Our institutional Fact Book does not disaggregate data on enrollment by major and race/ethnicity, providing this data publically only at the college level. However, as with the gender charts, there is no consistent effect on grades by underrepresented status.

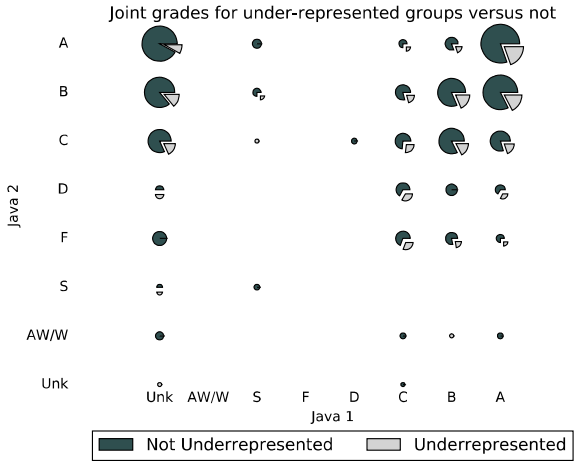

Fig. 7. Grades colored by self-reported ethnicity (broken into underrepresented and not underrepresented) for students who took both Java 1 and Java 2. The area of the pie chart is proprtional to the number of students with that joint grade.

Fig. 7 shows one area of potential concern about our academic support for students from underrepresented racial/ethnic groups, especially the groups with a C in Java 1 and a C, D or F in Java 2. These circles represent 10-15 students over a period of four years. This is an area that also could receive further investigation with a qualitative research methodology that is better suited to small sample sizes.

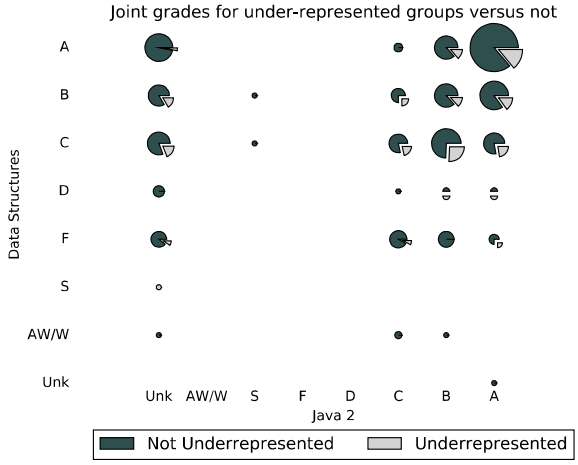

Fig. 8. Grades colored by self-reported ethnicity (broken into underrepresented and not underrepresented) for students who took both Java 2 and Data Structures. The area of the pie chart is proprtional to the number of students with that joint grade.

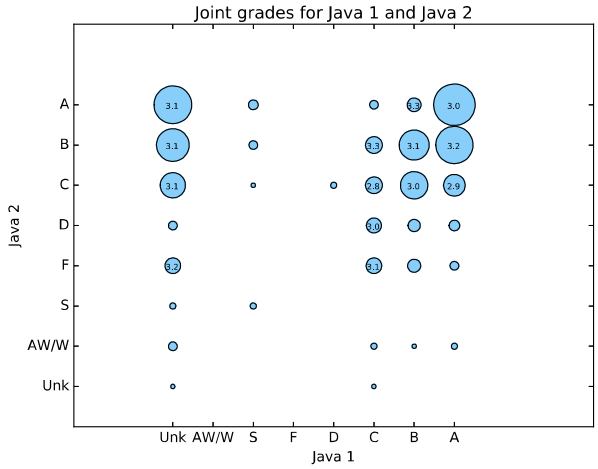

Fig. 9. Average GPA of students who took Java 1 and Java 2. The circle sizes and counts are as in Fig. 2 but the average GPA is displayed.

Figs. 6-9 show that our proposed prerequisite change does not have any known systematic bias against women and members of underrepresented racial/ethnic groups. This does not mean, of course, that it will not have unintended consequences in these populations—only that we were unable to anticipate those consequences based on historical data.

Fig. 3 and Fig. 4 provided strong evidence that success in the early CS classes was important for future success as a CS major. However, this success cutoff was implemented in our prior EMP as a GPA cutoff. Fig. 9 shows the mean GPA of the students shown in Fig. 1 and Fig. 10 shows the mean GPA of the students shown in Fig. 2. Mean grades range from 2.8 (C in Java 1, C in Java 2) to 3.7 (B in Java 2, C in Data Structures) with most of the larger groups having GPAs around 3.1. Interestingly, the students with As in Java 1 and Java 2 or Java 2 and Data Structures had mean GPAs of 3.0 and 3.3, respectively. We can see in Figs 3 and 4 that students with As in Java 1, Java 2 and Data Structures frequently end up as Computer Science majors. Neither graph shows a clear correlation between the students grades in the introductory CS courses and their OU Retention GPA, leading us to believe that OU Retention GPA is a poor indicator of success in CS courses. Combined with the evidence that GPA as a cutoff can negatively impact women and underrepresented groups [7], we have abandoned using GPA in all future discussions of managing enrollment. We also plan to share this work with other programs in our College that are using GPA-based thresholds for enrollment management.

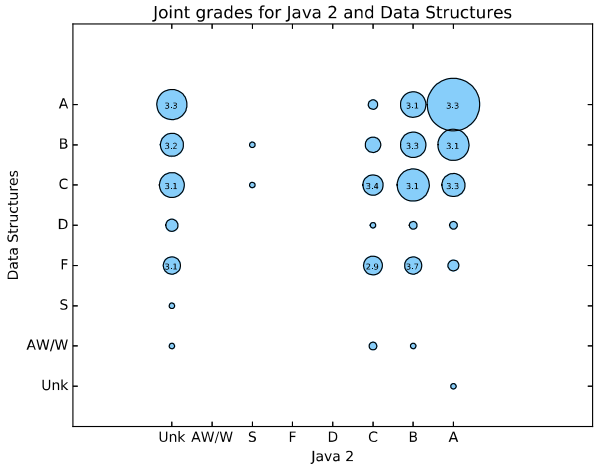

Fig. 10. Mean GPA of students who took Java 2 and Data Structures. The circle sizes and counts are as in Fig. 3 but the mean GPA is displayed.

IV. Assessment

While we found the initial analysis compelling, we wanted to assess it by individually examining the transcripts from past students who would not have been permitted to take data structures with this prerequisite in place. We found only 17 students from our four year data set fell into this category. We divided these students into five disjoint groups.

The first group of six students never attempted Data Structures or dropped the class in the first two weeks before a grade was recorded. Two of these students have left OU without graduating, while the others changed to majors that do not require Data Structures. Our change in policy will have no impact on these students. Similarly, these students will not reduce the size of the CS program as they left of their own accord.

The second group contains five students who had gone on to be successful as CS (2) or Computer Engineering (CpE) (3) majors. While not all of these students have graduated yet, all were successful in at least one upper division course within their major and are making reasonable progress in meeting graduation requirements. While closing enrollment to any student who could be successful is a concern, the historical record suggests that it is just one or two students a year. It should be remembered that students who do not meet the prerequisite for data structures would be allowed to repeat Java 2 to earn a higher grade under this plan. In fact, as this is now a requirement of the degree plan, students would be able to get financial aid support to repeat Java 2, something that is extremely difficult without this requirement. We hope that these students would make that choice. It also is possible that when students are aware of this change, they may work harder to get higher grades in the introductory classes the first time around, which may serve them better in general.

There were additional three groups with two students each. One group of students attempted Data Structures unsuccessfully (one twice, one three times—the maximum permitted by College of Engineering policy) and have since left OU without completing their degree.

A second group of two computer engineering students attempted Data Structures once, but withdrew from the course and delayed repeating it for a year or more. These students are, frankly, in more trouble than they may realize. Putting a year or more between introductory programming and data structures is not a good idea, especially for students who are struggling with programming. Because it is required for their CpE degree, they will have to take Data Structures during one of their final semesters at OU, probably concurrently with demanding engineering senior level electives and their capstone class, increasing the likelihood that a failure in Data Structures will delay or prevent their graduation. These students have other academic pathways they could have pursued. Electrical Engineering (in the same School as CpE at OU) has a broad overlap in requirements and only requires a single programming course for non-CS majors. If these students have to make this change three or four years into their academic program, it will be much more difficult. Being excluded from Data Structures may force these students to make a difficult but possibly necessary decision earlier.

The last group of two students are still CS majors, but have struggled to such a great degree that their graduation is in doubt. Both students have failed more than ten CS and Mathematics classes (D, F, W). Having students repeat so many classes does increases enrollment in CS. These students may have been better served by repeating Java 2 earlier, or finding a different major at an earlier time.

With this assessment data in hand, the Undergraduate Committee took our recommendation to the faculty of the School of Computer Science for the only assessment that matters: faculty approval of the prerequisite change. The faculty unanimously approved the prerequisite change after substantive discussion. This change has since been approved by the College of Engineering, the University Academic Programs Council, and the Provost (the final authority). The policy will be enacted in Fall 2017.

We have essentially completed advising and enrollment for the Fall 2017 semester, of the 141 students who are enrolled in Data Structures for Fall 2017, there were only five that did not meet the new prerequisite requirement. Two of these five students were computer engineering majors, one student was a math major (CS minor), and two were CS majors. After a substantial discussion, the Undergraduate Committee and the Director determined that these students would be permitted to enroll in CS 2413 both in Fall 2017 and Spring 2018 because it would be an unfair surprise to students who had completed Java 1 or Java 2 before this policy was enacted. This will delay the full implementation of the policy until Fall 2018. The Undergraduate Committee sent individual emails to each of the five students involved informing them of this research and suggesting (but not requiring) that they repeat Java 2 to have a better opportunity to pass Data Structures. None of the students has responded to the email at this time.

We considered waiting to publish this work until the policy had been in place for a year but decided against it because we felt that other programs facing enrollment challenges right now need to see our process. We are skeptical of whether waiting a year would provide more useful assessment data. We expect the number of students who leave CS because of this change to be small. The historical data suggests that it could be as little as four students a year, although increasing enrollment may bring in more students with fewer advantages and privileges and possibly more academic struggles. Students who leave an engineering program can be reluctant to participate in engineering education research. While Seymour and Hewitt had success in reaching this group [27], Meyer was only able to convince four students to be interviewed [28] from an entire College of Engineering.

V. Conclusions and Future Work

While this work has demonstrated that co-occurrences of course grades from transcript data can support decision making for managing enrollment, the size of the effect is predicted to be disappointingly small. It is possible that we could lose less than five students a year. This will not solve our current enrollment challenges. However, the approach of examining patterns of CS student grades in other courses, especially mathematics, may have a larger impact. It may be possible even to use mathematics placement scores to help guide students considering a CS major very early in their academic decision making.

The most important result from this research may well be the poor correlation between GPA and success in CS. At our institution, GPA based EMP are the norm. If other majors have the same pattern that ours does, a fact that has not been established, many departments may wish to find alternative ways of managing enrollment. The work of communicating our findings to other departments and possibly aiding them in making a similar analysis will begin shortly.

This plan is not without potential collateral damage. Both the Computer Engineering program and the Industrial and Systems Engineering with Analytics program rely on Java 1, Java 2 and Data Structures, among other CS courses. The faculty from these programs had no voice in the prerequisite change and their students were not included in the critical section of data analysis as neither program requires the CS pre-capstone course. CS minors also were not included, and in fact could not be included because minors at our institution are often first known to the department when they submit a request to minor shortly before graduation. It would be expected that this change might have a bigger impact on minors and the related majors than on CS majors, since presumably the students who are more interested in CS are CS majors. If this is the case, the impact on enrollment size may be much larger than our analysis anticipates.

We will never know the true size of the effect this change will have because student decision making processes are opaque. More than a decade ago the Undergraduate Committee discovered that none of our graduating students (from a one year sample) had a C in Java 1. We have told students in Java 1 that C grades in Java 1 can be problematic for people who plan to major in computer science for years. We have no idea how many students this statement discouraged from continuing the major. The new prerequisite may have more impact on decision making as students hear of others who are unable to take Data Structures without repeating Java 2.

This prerequisite change has a number of advantages over the prior GPA-base EMP. The most salient advantages are listed below.

- Students excluded from CS classes in the new plan will typically be leaving CS after taking only two CS classes, typically after the second semester of their first year or the first semester of their second year (the timing being largely determined by their mathematics placement). The prior EMP excluded students at the end of their second year although students with especially low GPAs sometimes left sooner. Changing majors sooner prevents students accumulating large numbers of credits that don't fit into a non-CS degree program and may help retain students in college.
- Since the new plan was based solely on student success, not on meeting specific enrollment goals, it can and should remain in place even if the CS enrollment takes an unanticipated downturn. The earlier EMP had to be removed immediately when enrollment decreased after the dot com bust. Making radical changes to GPA requirements (2.0 minimum GPA to 2.8 GPA back to 2.0 minimum GPA) is stressful for everyone, especially for students and the academic advisors.
- Changing course prerequisites requires no approval beyond the Provost level (a two to three month process). Changing GPA-based EMP requires approval at the Regent level (typically a six month process).
- The fact that grades from two classes are used in the prerequisite makes it more robust. We have hired adjuncts, taught online courses, or hired graduate students to teach Java 1 raising the possibility of some inconsistency in grading. Allowing students to get an A or B in one of two courses, diminishes the impact of inconsistencies in grading in Java 1.

It is a tragedy anytime that programs have to restrict enrollment, even more so when the criterion are arbitrary. Most students are in college only once, and will have only one major. To reject them from a major where they could be successful denies them the opportunity to have all of the advantages of that major. In the case of CS, this includes high paying jobs, plentiful employment opportunity, working for cutting edge companies, and the opportunity to understand deeply a discipline that is impacting virtually every aspect of human life on a daily basis. This is not something we should do lightly or randomly.

Acknowledgment

The authors thank Dr. John Antonio, Senior Associate Dean, Howard & Suzanne Kauffmann Chair in Engineering and Ms. Theresa Marks, Assistant Dean for Academic Student Services for providing the data used in this research.

REFERENCES

[1] Computing Research Association, Generation CS: Computer Science Undergraduate Enrollments Surge Since 2006. 2017.

[2] Wills, C. Analysis of Current and Future Computer Science Needs via Advertised Faculty Searches for 2017. CRA Bulletin, 2016.

[3] Zweben, S. and B. Bizot, 2015 Taulbee Survey: Continued Booming Undergraduate CS Enrollment; Doctoral Degree Production Dips Slightly, in Computing Research News. 2016, Computing Research Associates. p. 1-60.

[4] Basken, P. What Trump's Budget Outline Would Mean for Higher Ed. 2017 [cited 2017 April 27]; Available from: http://www.chronicle.com/article/What-Trump-s-Budget-Outline/239511.

[5] Camp, T., S. Zweben, E. Walker and L. Barker, Booming Enrollments: Good Times?, in Proceedings of the 46th ACM Technical Symposium on Computer Science Education. 2015, ACM: Kansas City, Missouri, USA. p. 80-81.

[6] Computer Research Association, Generation CS: Computer Science Undergraduate Enrollments Surge Since 2006. 2017.

[7] Patitsas, E., M. Craig and S. Easterbrook, A historical examination of the social factors affecting female participation in computing, in Proceedings of the 2014

conference on Innovation & technology in computer science education. 2014, ACM: Uppsala, Sweden. p. 111-116.

[8] Kaczmarczyk, L.C., A. Monge, J. Offutt, H. Pon-Barry and S. Westbrook. You should (and absolutely can) keep diversity in sharp focus during the enrollment surge. in 2015 Research in Equity and Sustained Participation in Engineering, Computing, and Technology (RESPECT). 2015.

[9] Public Affairs. OU Freshmen Fall to Spring Retention Rate Increases to 95.5%. 2015 [cited 2017 April 27]; Available from: http://www.ou.edu/publicaffairs/archives/OUFreshmenFallToSpringRetentionRateIncreases.html.

[10] Oklahoma State Regents for Higher Education, Policy and Procedures Manual. 2016.

[11] Margolis, J., Stuck in the Shallow End: Education, Race and Computing. 2008, Cambridge: Massachusetts Institute of Technology.

[12] Lareau, A., Unequal Childhoods: Class, Race, and Family. 2011, Berkeley, California: University of California Press.

[13] Ohland, M.W., et al., Race, Gender, and Measures of Success in Engineering Education. Journal of Engineering Education, 2011. 100(2): p. 225-252.

[14] Brawner, C.E., M.M. Camacho, S.M. Lord, R.A. Long and M.W. Ohland, Women in Industrial Engineering: Stereotypes, Persistence, and Perspectives. Journal of Engineering Education, 2012. 101(2): p. 288-318.

[15] Ohland, M.W., S.D. Sheppard, G. Lichtenstein, O. Eris, D. Chachra and R.A. Layton, Persistence, Engagement, and Migration in Engineering Programs. Journal of Engineering Education, 2008. 97(3): p. 259-278.

[16] Tyson, W., Modeling Engineering Degree Attainment Using High School and College Physics and Calculus Coursetaking and Achievement. Journal of Engineering Education, 2011. 100(4): p. 760-777.

[17] Redmond, K., S. Evans and M. Sahami, A large-scale quantitative study of women in computer science at Stanford University, in Proceeding of the 44th ACM technical symposium on Computer science education. 2013, ACM: Denver, Colorado, USA. p. 439-444.

[18] Walden, S.E. and C. Foor, "What's to keep you from dropping out?" Student Immigration into and within Engineering. Journal of Engineering Education, 2008. 97(2): p. 191-205.

[19] Academic Publications. OU General Catalog. 2017 8/9/2014 [cited 2017 April 27]; Available from: https://catalog.ou.edu/current/General_Info_Glossary.htm.

[20] Cohoon, J.P. and L.A. Tychonievich. Tapestry Workshop 2017. 2017; Available from: http://www.cs.virginia.edu/tapestry/.

[21] Cohoon, J.P., An introductory course format for promoting diversity and retention. SIGCSE Bull., 2007. 39(1): p. 395-399.

[22] Cohoon, J.P. and L.A. Tychonievich, Analysis of a CS1 approach for attracting diverse and inexperienced students to computing majors, in Proceedings of the 42nd ACM technical symposium on Computer science education. 2011, ACM: Dallas, TX, USA. p. 165-170.

[23] Barr, V. and D. Trytten, Using turing'scraft codelab to support CS1 students as they learn to program. ACM Inroads, 2016. 7(2): p. 67-75.

[24] Braught, G., T. Wahls and L.M. Eby, The Case for Pair Programming in the Computer Science Classroom. Trans. Comput. Educ., 2011. 11(1): p. 1-21.

[25] Margolis, J. and A. Fisher, Unlocking the Clubhouse: Women in Computing. 2002, Cambridge: The MIT Press.

[26] Institutional Research and Reporting. 2017 Fact Book. 2017 [cited 2017 April 27]; Available from: http://www.ou.edu/irr/fact-books.html.

[27] Seymour, E. and N.M. Hewitt, Talking About Leaving: Why Undergraduates Leave the Sciences. 1997, Boulder: Westview Press.

[28] Meyer, M. and S. Marx, Engineering Dropouts: A Qualitative Examination of Why Undergraduates Leave Engineering. Journal of Engineering Education, 2014. 103(4): p. 525-548.

.