An Evaluation of How Changes to the Introductory Computer Science Course Sequence Impact Student Success

Christine Reilly
Computer Science Department
University of Texas-Pan American
Edinburg, Texas 78539
Email: reillycf@utpa.edu

Emmett Tomai
Computer Science Department
University of Texas-Pan American
Edinburg, Texas 78539
Email: tomaie@utpa.edu

Laura M. Grabowski
Computer Science Department
University of Texas-Pan American
Edinburg, Texas 78539
Email: grabowskilm@utpa.edu

***Abstract*—Our university, as with many others throughout the world, has a relatively low pass rate in the introductory computer science courses. Over the course of more than fifteen years, various changes have been made to the introductory course sequence with the hope of improving student success. We describe these changes and perform an initial analysis of student course performance that finds little change in pass rates. We propose new changes to the course sequence and to individual courses within this sequence. These new changes are focused on increasing student engagement and developing the problem solving skills that are necessary for being a successful computer science major. We propose pilot projects that implement these changes and outline evaluation strategies for these pilot projects.**

I. Introduction

Over the span of more than fifteen years, our Computer Science department has made a number of changes to the sequence and content of the introductory courses. Many of these changes have been motivated by the faculty's desire to improve student success and learning outcomes, while other changes have been motivated by institutional factors. Our department is fairly typical in the fact that our courses change over time and with the motivating factors for these changes [1]–[5]. Various studies have found that computer science faculty tend to use intuition and anecdotal evidence as the motivation for their course changes. We acknowledge that many of our changes have been based on these non–scientific motivations, but now aim to evaluate the changes that have been made and to use research to drive future changes.

We begin this paper by describing the evolution of the introductory course sequence in our department over the past fifteen years. Next, we evaluate the effects of these changes on student success. We then discuss how other factors may impact the success of our students. Finally, we present changes that are in progress and outline our plans for evaluating the effects of these changes.

II. Overview of Our Computing Programs

Our university is a primarily undergraduate, state–supported institution with an enrollment of more than 20,000 students. We are a Hispanic Serving Institution with over 90% of these students being Hispanic. The majority of our students are first generation college students. The university primarily serves the local region, and the demographics of the university are similar to those of the region.

The Computer Science Department at our university offers an ABET accredited bachelor's degree in Computer Science, and master's degrees in Computer Science and Information Technology. The department also offers an ABET accredited bachelor's degree in Computer Engineering, in cooperation with the Electrical Engineering Department. In Fall 2014, we had 343 undergraduate Computer Science majors, 282 Computer Engineering majors, and 124 master's students.

As is typical at many universities throughout the world [6], our sequence of introductory computer science courses have a high fail rate. Over a four and a half year period, we found that on average both our CS1 and CS2 courses had a 60% pass rate [7]. The instructors who teach these courses are frequently examining methods for improving the pass rates. In addition to making changes to the individual courses, we also consider these courses as a sequence. In this paper, we focus on the pipeline of introductory computer science courses as a sequence and examine how changes in this sequence impacts student performance. We begin this discussion with a presentation of the recent history of our introductory computer science course sequence.

A. History of Computer Science Introductory Courses

When the department was formed in the mid-1990's we initially offered a three–hour CS1 course that focused on introducing programming concepts and a three–hour CS2 course that provided an introduction to data structures. The core objectives of our CS1 and CS2 course remain largely the same today as they were almost 20 years ago. One change is that now CS1 is a four credit–hour course (three hours of lecture, and one credit–hour for a 3 hour seat time lab). The major changes during the past 20 years have been to the course sequence, as described in detail below. These changes included the introduction of a course in-between CS1 and CS2, then the removal of this course along with the introduction of a CS0 course.

In 1998, a three–hour course called "Foundations of Computer Science" (course number CSCI 1381 at our university) was introduced and was placed in between CS1 and CS2 in

the course sequence. One of the goals of this course was to include the breadth of computer science early in the sequence of courses for Computer Science majors, while another goal was to help students better transition to the level of abstraction required for CS2. The impacts of CSCI 1381 on student learning outcomes and success in CS2 were analyzed after CSCI 1381 had been offered for three years. It was found that students reported that they achieved the learning outcomes, but there was not a clear improvement in performance in CS2 [8]. Some of the lack of improvement in CS2 performance was attributed to students not taking the courses in the proper sequence. Additionally, the impacts of CSCI 1381 likely changed over time because the number of programming assignments in CSCI 1381 were increased following the first few semesters that the course was offered.

In the late 2000's, the Computer Engineering program began offering a Bachelor's degree in Computer Engineering. This program is jointly administered by the Computer Science and Electrical Engineering departments. Most of the Computer Engineering courses are cross–listed with either Computer Science or Electrical Engineering courses. The Computer Engineering majors take the same CS1 and CS2 courses as Computer Science majors. However, CSCI 1381 was not included in the Computer Engineering degree program. Therefore, upon reaching CS2, the Computer Science and Computer Engineering majors had different levels of programming experience.

In Spring 2011, a one–hour "Introduction to Computer Engineering" (course number CMPE 1101) was introduced as a requirement for the Computer Engineering major and was set as a pre–requisite for CS1. The goals and content of CMPE 1101 are discussed below. One of the reasons for introducing CMPE 1101 was to more closely align the Computer Engineering major to the other engineering majors within our college that already had a one–hour introductory course.

The final change to the introductory course sequence came in Fall 2011 when CSCI 1381 was phased out and a one–hour "Introduction to Computer Science" (course number CSCI 1101) was introduced. This change was made to more closely align the beginning of the Computer Science major courses with the Computer Engineering major courses.

We also note that some of our Computer Science and Computer Engineering students have transferred into our university from the local community college. The community college offers a three–hour version of CS1 and continues to offer an equivalent course to CSCI 1381. However, the community college does not offer a course that is equivalent to CSCI 1101 or CMPE 1101. For this study, we focus on the students who began the introductory computer science course sequence at our university, but we acknowledge that some of the students in our CS2 course took CS1 and CSCI 1381 at the local community college.

B. Description of Our CS0 Courses

Currently, the first class in the sequence of introductory computer science courses for students who major in Computer Science is "Introduction to Computer Science" (CSCI 1101). The first class in the sequence of introductory computer science courses for students who major in Computer Engineering is "Introduction to Computer Engineering" (CMPE 1101). CMPE 1101 was first offered in Spring 2011, and CSCI 1101 was first offered in Fall 2011. Initially, the courses were very similar and they remain mostly similar to this day.

TABLE I. TOPICS COVERED IN CSCI AND CMPE 1101

Number representation • Unsigned binary numbers • Hexadecimal representation • Signed binary numbers • Binary floating point representation
Using bits to represent text, images, and sound
Digital logic • Boolean algebra • Circuits
Computer architecture, machine language, and program representation
Algorithms and problem solving • Programming using LEGO Mindstorms • Overview of searching and sorting algorithms
Miscellaneous topics • Operating systems • The Internet and the World Wide Web

The prerequisite for CS1 is either CSCI 1101 or CMPE 1101, along with a co-requisite of college algebra. Both CSCI 1101 and CMPE 1101 are one credit–hour laboratory courses that have two and one half hours of seat time per week. They are typically offered on a two–day per week schedule for one hour and fifteen minutes per day.

CSCI 1101 and CMPE 1101 were started following a similar pilot course [9]. Much of the computer science breadth material from CSCI 1381 was carried over to CSCI 1101 and CMPE 1101. These courses also include programming assignments using the LEGO Mindstorms robots. The LEGO Mindstorms programming environment has an intuitive drag and drop block programming interface. This environment allows students to gain experience with basic programming concepts and control structures without the frustration of syntax errors. The goal of the programming component of CSCI 1101 and CMPE 1101 is to provide an introduction to programming concepts so that students already have some familiarity with these concepts when they reach CS1.

Table I shows an outline of the topics that are covered in both CSCI 1101 and CMPE 1101. The Computer Science course tends to spend a bit more time on algorithms and programming, while the Computer Engineering course tends to spend a bit more time on number representation.

The class time for both CSCI 1101 and CMPE 1101 is approximately evenly split between traditional lecture and time that students spend working on the lab assignments. Because these are scheduled as laboratory classes, the goal is for students to be able to complete the bulk of the coursework during class time.

A typical section of CSCI 1101 or CMPE 1101 has between twenty and forty students. Because the class is held in a computer lab, enrollment is absolutely capped at a maximum of 45 students. In addition to the full–time faculty instructor,

TABLE II. PASS RATE IN CS1 BASED ON WHEN 1101 IS TAKEN

When Took CMPE or CSCI 1101	Number of Students	Percent Passing CS1
Never took 1101	397	60%
Prior to CS1	155	54%
Concurrently with CS1	163	63%
After CS1	87	55%

CMPE 1101 typically has one or two graduate student teaching assistants assigned to each section in order to assist during the lab. Typically, CSCI 1101 has a full–time faculty instructor plus an undergraduate student assistant.

III. IMPACT OF CHANGES TO COURSE SEQUENCE

We have performed an initial data analysis in order to evaluate whether or not taking CSCI 1101 or CMPE 1101 helps students perform better in CS1. This analysis is summarized in Table II. The data set includes all students who took CS1 at our university between Fall 2009 and Fall 2014, a total of 802 students.

The column titled "When Took CMPE or CSCI 1101" indicates when the student took the 1101 course in relation to when they took CS1. The first row, "Never took 1101" is likely to mostly contain students who took CS1 prior to the introduction of 1101. Students in the row labeled "Prior to CS1" are those who took 1101 in some semester prior to the semester when they took CS1. The students labeled "Concurrently with CS1" are those who took 1101 and CS1 in the same semester. The last row, labeled "After CS1" are students who took 1101 in a later semester than they took CS1.

The results shown in Table II are somewhat counterintuitive and we hypothesize that the results are indicating the impact of factors other than the 1101 course on student performance in CS1. These initial results show that students who take the 1101 course during the same semester as they take CS1 have a higher pass rate in CS1 than the students who take 1101 prior to taking CS1. The Computer Science department and Computer Engineering program are typically willing to give permission for students to take 1101 and CS1 concurrently for those students who have prior programming experience or have other indicators that they are well prepared for CS1. Therefore, the students who are taking 1101 and CS1 concurrently are likely to be higher performing students in general.

Another observation of the data presented in Table II is that the pass rate in CS1 for students who never took 1101 is not remarkably different from the pass rate of students who did take 1101. One of the goals of having a CS0–type course is to better prepare students for CS1. It does not appear that our 1101 course is currently meeting that goal.

At this point, we could dive deeper into the data that we have on grades in these courses. We could also analyze whether removing CSCI 1381 (the course that was once taken between CS1 and CS2) has had any impact on student performance in CS2. However, our intuition is that there are other factors that are having a greater impact on the success of students in the pipeline of introductory computer science courses than a simple analysis of performance from one course to the next can reveal. Also, we think it is unlikely that we could reintroduce a course in–between CS1 and CS2. Therefore, we have decided to focus the remainder of this paper on taking a closer look at some of the other factors that may influence the success of students in our courses, as well as considering how we could better leverage our current courses to better meet the educational needs of our students.

IV. IMPACT OF OTHER STUDENT SUCCESS FACTORS

As was presented in Section II, the majority of students at our university are first generation college students who attended high school in the Rio Grande Valley region of South Texas. This region has one of the highest poverty rates in the United States, and is also one of the most rapidly growing regions of the country. Students enter our university with a wide range of levels of college readiness.

The faculty in our department have informally noted that many of the students who do not pass our courses are not fully engaged in the course. These students typically have poor class attendance and do not complete all of the required work for the course. Of course, if a student does not turn in the required course work, they will not earn the points necessary for passing the course.

In this section, we present a snapshot of the amount of coursework that is completed by students who fail CMPE 1101 and CS1. We also reflect on various issues that affect student engagement. In Section V we will discuss ideas for changes that may increase student engagement.

We compiled data on the percent of assigned coursework that was completed by the students who failed CMPE 1101 and CS1 over a number of semesters. In CMPE 1101, this assigned coursework is the lab assignments that are worth 50% of the course grade. Figure 1 shows the fraction of the students who failed CMPE 1101 (on the y–axis) that completed various amounts of the required work (on the x–axis). CS1 has two types of assignments. The lab assignments, which are intended to be completed during the weekly lab period, are worth 30% of the course grade. There are also larger programming assignments that are completed throughout the course of the semester and are worth 20% of the course grade. Figure 2 shows the fraction of the students who failed CS1 (on the y–axis) that completed various amounts of the labs and programming assignments (on the x–axis). We see in Figures 1 and 2 that a few of the students who fail do complete all or almost all of the required work. However, the majority of students who fail these courses are completing a small amount of the required coursework.

We understand that there are many factors that may impact an individual student's decision of whether or not to complete required coursework. Because most of the students in these courses are lower–division students, they may have not yet developed good time management skills. From conversations with our students, we have learned that some of them are balancing other demands on their time in addition to school. Some students work more than 20 hours per week, or have family responsibilities that require a good amount of their time. The combination of work and family commitments with poor time management skills can make it difficult for a student to complete the coursework. Another factor that may affect many of our students is that first–generation college students are sometimes hesitant to ask for help, may not recognize when

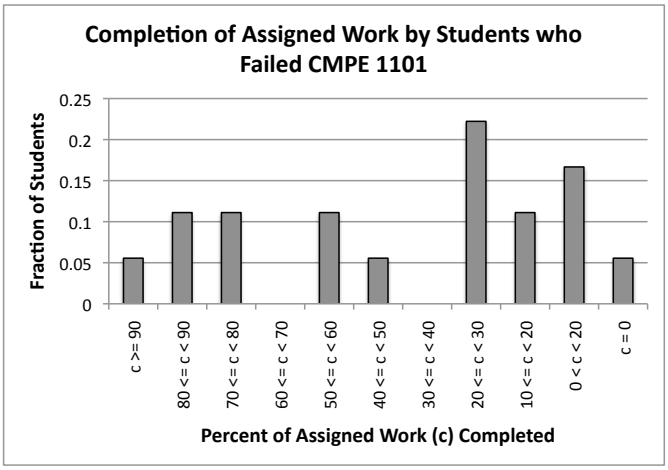

Fig. 1. Completion of Assigned Work by Students who Failed CMPE 1101

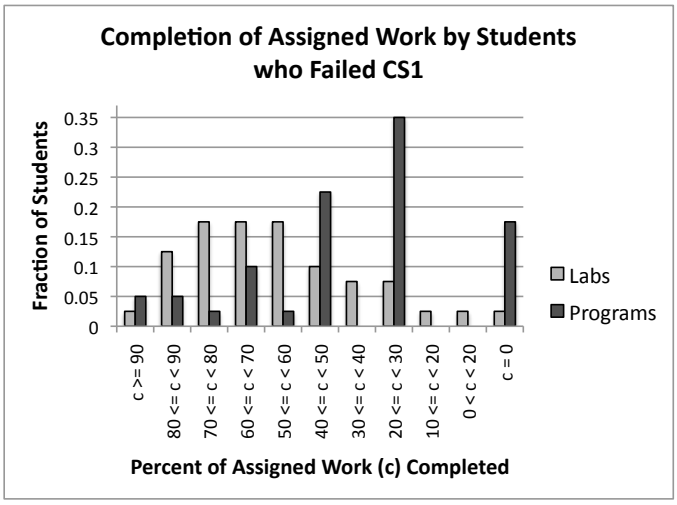

Fig. 2. Completion of Assigned Work by Students who Failed CS1

they need to ask for help, or may not be aware of the various sources of help that are available.

Therefore, we propose that the most effective approach for improving performance in our introductory computer science courses is to focus on developing and encouraging a collaborative atmosphere that provides many opportunities for students to become engaged in the learning process. We also propose the need to focus on the overall concepts and skills that are required for computer science and computer engineering, as opposed to focusing on small details.

V. Ideas for Moving Forward

Over the past several years, the Computer Science faculty have often discussed how we could redesign the courses in the introductory sequence to improve student success. Suggested changes involve revisions to the course sequence as a whole and to individual courses. The various ideas for changes demonstrate how the Computer Science faculty continue to struggle with identifying the core issues that affect student success in Computer Science and Computer Engineering courses.

During many informal discussions, our Computer Science faculty have identified common issues among students that appear to lead to diminished student success in computer science, both at the introductory level and downstream. These issues include a general lack of problem solving skills, low engagement and motivation, lack of participation in class and in completing assignments out of class, and difficulty with varying levels of abstraction. These issues are certainly not unique to our department, university, or program. Many university faculty members express concerns about student engagement [1], and the growing emphasis on active learning reinforces the importance of drawing students into an active learning process.

In this section, we first describe ideas for changes to the sequence of introductory computer science courses. Then we discuss specific changes to the CS0 and CS1 courses that are currently being piloted or will be piloted in an upcoming semester.

A. Changes to the Pipeline

Several of our faculty members advocate reviewing the entire introductory course sequence (CSCI or CMPE 1101, CS1, CS2) by first identifying what the *sequence* should accomplish, as opposed to thinking about the courses in isolation. This type of revision is far more sweeping than the incremental changes that have been made over time by individual faculty members. Consequently, such alterations to the introductory sequence have not yet been implemented.

One suggested approach to these far-reaching revisions to our introductory sequence focuses on improving engagement and problem solving skills. With this approach, CS0 (*i.e.*, CSCI and CMPE 1101) will focus on problem solving and broad ideas, providing a better foundation for programming in CS1 and CS2. Some details of proposed changes to the content of CS0 are provided below. In this redesign, the CS1 course will concentrate on enhancing problem solving skills and introducing core programming concepts. Even though the existing CS1 course content includes all the core programming concepts, some faculty members feel that the students struggle with the details of syntax in C++. It is also challenging for faculty to devise interesting and engaging assignments for CS1 using C++. In light of these two notions, we are considering changing the language used in CS1 to Python, with a switch to C++ in CS2. This approach has been tested successfully at other institutions, with no significant difference in performance between students who switched languages after CS1 and those who used C++ in both CS1 and CS2 [10]. Our reasons for considering using Python for CS1 include the relative ease of use and reduced syntactic complexity of Python as compared to C++, and the fact that the many libraries available in Python enable students to execute more interesting and engaging programming projects much sooner than students writing in C++. In this revised course sequence, CS2 will begin with a brief review of the CS1 content to introduce C++ to the students, and then continue with much of the current CS2 content (*e.g.*, dynamic memory allocation, data structures), retaining a focus on modular programming, with nearly all considerations of object-oriented programming and design addressed in an additional course. We have not yet fully addressed how to deal with the impact of the proposed redesign

on downstream computer science courses, but a number of possible solutions to these issues have been suggested and we are confident that this is not a major stumbling block to pursuing the revision of the introductory sequence.

B. Changes to CS0

Although the Computer Science department faculty have not yet determined how extensively the introductory course sequence should be modified, two faculty members who teach CS0 are working on revisions to that course that can set the stage for the larger revisions discussed above. The two faculty members will collaborate on developing a new course structure and course materials for CS0 in Summer 2015, and will use the revised course in two pilot sections in Fall 2015. We hope to leverage existing collaborations with colleagues in social sciences and education to design tools to measure the impact of the revised course on student problem solving skills and student engagement.

The modified course will revolve around three "big" ideas: algorithms, the stored program concept, and social impacts of computing. These three ideas provide a conceptual framework for organizing the course content, and allow exploring ideas that students will revisit time and time again in their future course of study. The updated course will use a strong active learning approach that encourages and rewards student involvement. While these approaches and activities are not novel, introducing more active learning in our CS0 course will set the stage for increased use of such approaches throughout courses in the Computer Science department.

Algorithms is, without a doubt, a core concept in computer science and engineering. Beginning the CS0 course with algorithms allows us to address topics such as problem solving processes and paradigms, algorithm design and description, and the notion of "one algorithm, many implementations, many languages." These ideas will be supported with hands-on class activities. Problem solving approaches will be investigated through games and puzzles. Creating and communicating an algorithm clearly can be explored in engaging activities such as building with Legos. The concept that one algorithm can be executed many ways and with many programs will be addressed by exercises such as implementing the same simple algorithm using different tools (*e.g.*, Scratch and Lego Mindstorm robots). Placing algorithms at the start of the course introduces important concepts, but also may promote student interest and increase engagement in the course. With this topic, we have the opportunity to have students do many game-like activities early on and have them working with robots right away in the course.

The stored program concept is critical to both programming and the design of computing machinery. This core idea allows us to address issues such as data representation, the instruction cycle, and computer hardware components. Although this topic appears much "drier" than the previous topic at first glance, there are many activities available that deal with the information in appealing ways.

CS0 provides an ideal context for introducing the idea of social impacts of computing. Although our students will all take professional ethics courses as part of their degree requirements, addressing the societal and cultural dimensions of computing in the very first major course introduces a discipline-centered perspective on those questions and underscores the importance of considering these issues. This topic will allow us to address issues related to privacy, security, and professional ethics in computer science and engineering. We will also be able to explore sustainability as it relates to computing. This topic is a particularly good opportunity to provide content that is tailored for computer science or computer engineering students. For example, the computer engineering students may look at considerations in chip design that impact power consumption, while computer science students may explore a variety of software applications that support community sustainability and green technology.

C. Changes to CS1

In academic year 2014 – 2015, we began a pilot study in some sections of our CS1 course with the goal of identifying more effective interventions for student success. These interventions are part of a large–scale collaboration between the College of Engineering and Computer Science and the College of Science and Math at our university. The first initiative is the development of a pre–test for CS1 that could identify students who are at risk of failing. The second initiative is the use of the pre–test data to inform intentional intervention with peer mentors. An overview of these initiatives is provided below. We plan to publish a separate paper that details the results of these interventions following the conclusion of the current study.

While the pre–test data is only preliminary, we have already found interesting results that are driving the continued development of the pre–test. Through two semesters, pre–test questions on basic algebra and logic puzzles have had virtually no correlation (correlation coefficient < 0.1) with exam scores or final course scores. Samples of these questions are shown in Table III. This was quite surprising, as the connection between these questions and CS1 material seems intuitively obvious. One question that does have a correlation with exam scores or final course grades is a multi–equation word problem, which had a correlation coefficient of 0.28 in the first semester ($n = 31$) and 0.173 in the second semester ($n = 41$). This problem is shown in Table IV.

In the second semester, we tried pre–test questions on simple automation, showing a robot on a grid with pseudo–code to move it up, down, left, or right with conditional checks for blocked squares. We also included a question requiring students to post a simple note on the course blog later that day, in order to examine organizational skills. Those questions showed more promise, with an overall correlation coefficient of 0.30 ($n = 41$).

The second intervention we are piloting in some CS1 sections is peer mentoring. An often overlooked element of engaged learning is having effective life skills, including a good social support network. A student can be interested and have every intention of learning, but fail to manage the process and fall short in their learning goals. This is particularly true for lower–division students who are adjusting to college life. For this reason, the mentoring initiative in this course is focused on relationships, goals, plans, and progress rather than traditional tutoring. Students met, in groups, with a peer

TABLE III. CS1 PRE–TEST QUESTIONS THAT DO NOT CORRELATE WITH COURSE PERFORMANCE

1)	Given the equation: $x = 8 \times (y + 3)$ Let $y = 2$, then the value of x is ____
2)	Given the equation: $y = 8 \times 2(x - 1)$ Let $y = 6$, then the value of x is ____
3)	Five cities got more rain than usual this year. The five cities are: Last Stand, Mile City, New Town, Olliopolis, and Polberg. The cities are located in five different areas of the country: the mountains, the forest, the coast, the desert, and in a valley. The rainfall amounts were 12 inches, 27 inches, 32 inches, 44 inches, and 65 inches. • The city in the desert got the least rain; the city in the forest got the most rain. • New Town is in the mountains. • Last Stand got more rain than Olliopolis. • Mile City got more rain tha Polberg, but less rain than New Town. • Olliopolis got 44 inches of rain. • The city in the mountains got 32 inches of rain; the city on the coast got 27 inches of rain. Which city got the least amount of rain? Where is Mile City located?

TABLE IV. CS1 PRE–TEST QUESTIONS THAT DO CORRELATE WITH COURSE PERFORMANCE

John is three times as old as Greg, and Greg is half the age of Bob. Steve is two times the age of John and Bob combined. If Steve's age is 60, how old is Greg's older cousin Jane, who is 2 years older than Greg? Jane's age: ____

mentor to discuss their expectations, methods of preparation, and results from the first 2–week exam in the course. Each student was required to create a plan of study to go over with the group indicating their expectations and actual experiences with time spent in different course activities. While this is initiative is in its early stages, we have some evidence that students connected with each other for support networks and followed up with the mentor for tutoring at a higher rate. Going forward with this activity, we are developing an online web–based tool for students to do simple weekly progress check–ins, so that they can see how their expectations and effort correlate to performance in class activities. The peer mentors in academic year 2015 – 2016 will be working to roll out that system with the students and evaluate its effectiveness in keeping them engaged and actively planning for success.

VI. CONCLUSIONS

This paper marks a shift in the focus we are taking when making changes to our introductory computer science courses. Instead of focusing on small performance metrics, we are now looking at the bigger picture of student success. The changes that we are piloting in select sections of our CS0 and CS1 courses aim to increase student engagement and confidence.

We expect that our introductory computer science courses and course sequence will continue to evolve over time. Our goal is to make changes that are motivated by well–known best practices in engineering education and to scientifically evaluate the changes that we make. We are currently working to achieve this goal by using pilot sections to test changes and using initial results from these pilot projects to guide future changes.

REFERENCES

[1] L. Barker and J. Gruning, "The student prompt: Student feedback and change in teaching practices in postsecondary computer science," in *Proceedings of the 2014 Frontiers in Education Conference*, Madrid, Spain, October 2014.

[2] L. Barker, C. L. Hovey, and J. Gruning, "What influences cs faculty to adopt teaching practices?" in *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, Kansas City, MO, USA, March 2015.

[3] S. Fincher, B. Richards, J. Finlay, H. Sharp, and I. Falconer, "Stories of change: How educators change their practice," in *Proceedings of the 2012 Frontiers in Education Conference*, 2012.

[4] D. Fossati and M. Guzdial, "The use of evidence in the change making process of computer science educators," in *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education*, Dallas, Texas, USA, March 2011.

[5] L. Ott, "Explorations in computing: Could this be the key to retention?" in *Proceedings of the 2014 Frontiers in Education Conference*, Madrid, Spain, October 2014.

[6] J. Bennedsen and M. E. Caspersen, "Failure rates in introductory programming," *SIGCSE Bulletin*, vol. 39, no. 2, pp. 32–36, June 2007.

[7] C. F. Reilly and E. Tomai, "An examination of mathematics preparation for and progress through three introductory computer science courses," in *Proceedings of the 2014 Frontiers in Education Conference*, Madrid, Spain, October 2014.

[8] P. Brazier, L. Grabowski, and G. Dietrich, "Closing the CS I - CS II gap: A breadth-second aproach," in *Proceedings of the 2003 Frontiers in Education Conference*, 2003.

[9] L. M. Grabowski and P. Brazier, "Robots, recruitment, and retention: Broadening participation through CS0," in *Proceedings of the 2011 Frontiers in Education Conference*, 2011.

[10] R. J. Enbody, W. F. Punch, and M. McCullen, "Python CS1 as preparation for C++ CS2," *SIGCSE Bull.*, vol. 41, no. 1, pp. 116–120, March 2009.