Designing an Introductory Programming Course to Improve Non-Majors' Experiences

Jessica Q. Dawson
University of British Columbia
jqdawson@cs.ubc.ca

Meghan Allen
University of British Columbia
meghana@cs.ubc.ca

Alice Campbell
University of British Columbia
alicecam@cs.ubc.ca

Anasazi Valair
University of British Columbia
sazi.valair@ubc.ca

ABSTRACT

Demand for computing courses from students in disciplines outside of Computer Science is growing. This growth has created increasing challenges in offering one-size-fits-all CS1 courses. We found that non-CS majors' experiences and outcomes in our existing CS1 course were worse than those of intended CS majors. In response, we developed an introductory programming course, CS0.5, aimed at meeting the needs of the diverse population of non-CS major students interested in our courses. In this paper, we present the motivation, curriculum design, and evidence of effectiveness for this new course. We describe the specific design decisions we made in response to the experiences of non-CS majors in CS1. We also demonstrate that students' outcomes in CS0.5–measured in terms of students' pass rates, satisfaction, and attitudes–all not only improve compared to non-CS majors in CS1, but also largely match those of CS majors in CS1. Finally, we present student feedback, gathered through surveys and Appreciative Inquiry focus groups, that illustrates how our curriculum design choices better meet our non-major students' needs. The most-valued course design elements, as identified by focus group participants, provide insight for other CS educators who are designing similar courses.

CCS CONCEPTS

• **Social and professional topics → Computer science education**;

KEYWORDS

CS for all; non-majors; curriculum design; CS1; blended-learning; evaluation; appreciative inquiry

ACM Reference Format:
Jessica Q. Dawson, Meghan Allen, Alice Campbell, and Anasazi Valair. 2018. Designing an Introductory Programming Course to Improve Non-Majors' Experiences. In *SIGCSE '18: The 49th ACM Technical Symposium on Computer Science Education, Feb. 21–24, 2018, Baltimore, MD, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3159450.3159548

SIGCSE '18, Feb. 21–24, 2018, Baltimore, MD, USA

ACM ISBN 978-1-4503-5103-4/18/02…$15.00
https://doi.org/10.1145/3159450.3159548

1 INTRODUCTION

Computer Science departments have seen dramatic increases in enrollments. This growth has been driven by increasing numbers of CS major enrollments, but also by increasing demand for computing courses from non-CS majors who recognize the mounting importance and relevance of computing in their own fields [4, 17]. At our institution, this growth has created new challenges to providing introductory programming courses that fit the needs of all our students. As our student population expands, we want to ensure we offer courses that support diversity in computer science and that students have positive experiences in our courses regardless of their academic area of interest.

Prior to 2016, our university offered only one first-year, introductory programming course (CS1). It is 4-credits, and serves approximately 1500 students per year, including students who intend to be CS majors, students who are required to take the course for their major, and students who are taking the course as an elective. In our CS1 course, which is based on the How to Design Programs curriculum [9], students learn to systematically design readable, well-structured, and well-tested programs using functional teaching languages. While the course has been effective and student feedback has been largely positive, anecdotally, course instructors reported that non-CS majors had worse outcomes, and were more likely to say that they did not enjoy the course, that the pace was too fast, and that the workload was too high.

To serve these students, we proposed a new, first-year, computer science class (CS0.5) about systematic program design for non-major students. Our approach to the CS0.5 curriculum design was driven by the assertion that different disciplines see the world differently [14]. In CS1, we emphasize the core programming and software design skills and concepts that are necessary for computer scientists, but for non-majors we decided that it was more useful to focus on how programming can be used in their own academic discipline. We aimed to shift the emphasis without compromising the integrity of the authentic programming and software design skills that students learn in CS1, particularly as non-majors may only take one computing course.

CS0.5, launched in September 2016, is a 3-credit course that uses a blended-learning approach and is taught in Python. It covers about 40% of the learning outcomes of CS1 over six modules, CS0.5-specific learning outcomes over two modules, and a CS0.5-specific project that is designed so that each student can apply their program design skills to their own area of interest.

In this paper, we report on our experience designing, launching, and evaluating the first two semesters of CS0.5. First, we report on an investigation that examined the differences between CS and non-CS major students' pass rates in and satisfaction with CS1. We also report differences in CS1 students' attitudes towards computing that we gathered using a validated survey tool [7]. In the context of related work, we describe the CS0.5 curriculum and the design decisions we made in response to the experiences of non-CS majors in CS1. We demonstrate that students' outcomes–measured in terms of pass rates, satisfaction, and attitudes–all improve with CS0.5 when compared with non-CS majors in CS1. We present student feedback, gathered through surveys and Appreciative Inquiry [19] focus groups, that illustrates how our curriculum design choices effectively target non-major students' needs. Finally, we discuss student-identified areas of improvement for CS0.5.

2 MAJORS AND NON-MAJORS IN CS1

In 2015/2016, we conducted an evaluation of CS1 to help us better understand the non-CS major population of the course[1]. We wanted to identify similarities and differences between the non-major students' experiences and course outcomes, and those of intended CS majors, in order to guide our development of CS0.5.

2.1 Pass, fail and withdrawal rates

We examined the proportion of students who passed CS1 in 2015/2016 Term 1 and Term 2 (n = 1225; 39% female[2]). We included in the enrollment total all undergraduate students who withdrew within the first two weeks, which is the normal deadline to drop a course without it appearing on their transcript. Overall, 74% of all students passed the course, with 19% failing and 7% withdrawing.

At our institution, most students don't declare their major until after first year, and so we are unable to distinguish intended majors from non-majors using the enrollment data available. As a proxy for comparing CS majors and non-majors, we used the most common faculties that offer majors in CS: Science, Arts, and Commerce[3]. Students from other faculties or schools, such as Forestry or Education or Engineering, also enroll in CS1, but in much smaller numbers.

We found that Science and Commerce students had better pass rates than the overall average, both at 81%. This is 10% higher than for students in Other programs, of whom 71% passed, and 21% higher than for students in Arts, of whom only 60% passed. At our university, the majority of students who major in CS are in the Faculty of Science, although many Science students who take CS1 are also non-CS majors. Roughly two thirds of the Commerce students in CS1 in 2015/2016 were in a combined CS and Business program. A much smaller proportion of CS majors are in the Faculty of Arts. Thus, the lower pass rates for students in Arts and Other programs suggest that non-CS majors were disproportionately more likely to fail or withdraw from CS1.

[1]All evaluations were approved by our institution's Behavioural Ethics Review Board.
[2]Our gender data is from our student information system, which presents gender as binary, and does not accurately represent students with non-binary gender identities.
[3]We also offer a second degree program in computer science for mature students with previous degrees (n=79 in CS1 2015/2016). These students enter as CS majors, and are required to take the existing CS1 course with their cohort. These students are not as useful as a comparison group with non-major students, and have been excluded. We also excluded graduate students, who number only one or two per year.

2.2 Student feedback from surveys

At the end of each term we conducted an online survey to gather information about students' CS1 experiences, which we used to inform the CS0.5 course design. The survey was part of a larger evaluation that extended over the term, and it included a range of questions aimed at course improvement. For this paper we were most interested in exploring the differences between students at the end of the course as opposed to individual shifts from beginning to end. Thus, we present here only results from the end-of-term survey and only questions that relate to the measures of interest in this paper, namely satisfaction and attitudes towards computing.

A total of 626 (55%) of the 1134 students registered at the end of term participated. We asked respondents to specify their intended major(s): those who chose CS as an intended major were classified as CS majors (50%) and all others as non-CS majors (50%). We used unpaired Welch's t-tests to compare the satisfaction and attitudes of CS majors and non-CS majors. See Figures 1 and 2; all questions were optional, so the n varies for each group and question.

Satisfaction: We asked students to rate their satisfaction with various elements of the course. We found that non-CS majors were significantly less satisfied than CS majors with their overall experience, their expected grade, the time and effort that they put into the course, and the skills and knowledge that they learned.

Attitudes towards computing: We asked respondents a subset of questions from the Computing Attitudes Survey (CAS) [7], a validated tool to measure students' expert-like attitudes towards computing. It consists of statements on which there is an empirically-established expert opinion, and which cluster into five empirically-determined subscales measuring different facets of students' attitudes. Respondents rate their agreement with each statement on a 5-point likert scale (strongly agree to strongly disagree).

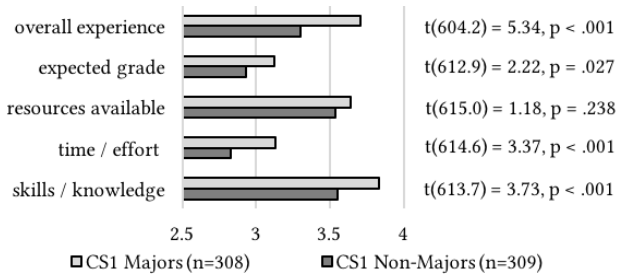

Figure 1: Unpaired t-test of satisfaction in CS1 (2015/2016); 1=very unsatisfied; 5=very satisfied.

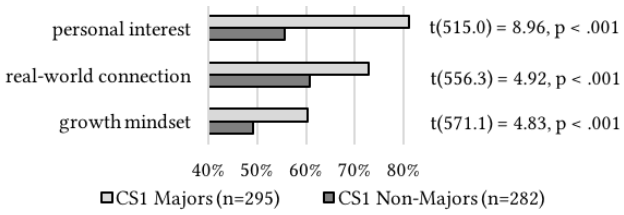

Figure 2: Unpaired t-test of students' agreement with expert opinions on CAS factors in CS1 (2015/2016).

We used a subset of the statements from the three subscales most related to the attitudes we wanted to foster in non-majors. The *real-world connections* subscale (4 statements) focuses on relationships between computing and the real world, e.g. "I think about the Computer Science that I experience in everyday life," where the expert opinion is agreement. The *personal interest* subscale (3 statements) focuses on students' enjoyment and engagement with computing, e.g. "I enjoy solving Computer Science problems," where the expert opinion is agreement. The problem-solving fixed mindset subscale (7 statements), which we call *growth mindset*, focuses on students' confidence and mindset when solving computing problems, e.g. "If I get stuck on a computer science problem, there is no chance I'll figure it out on my own," where the expert opinion is disagreement.

We scored the students' responses to the CAS statements following the standard procedure [7]. We collapsed the responses into a 3-point range (disagree, neutral, agree), and then compared each student's response to the established expert opinion; a student's score on each subscale is calculated as the proportion of statements belonging to that subscale where the student's response matched that of the experts. For each factor, a higher percent agreement suggests more expert-like attitudes, i.e., more personal interest, a stronger appreciation of the relationship between computers and the real world, and more confidence and a more growth-oriented mindset when solving problems. We compared the scores for each group using Welch's t-tests, and found that non-CS majors had significantly less expert agreement than CS-majors on all subscales.

Student comments: To better understand non-CS majors' experiences in CS1 and the reasons for the differences in satisfaction and attitudes towards CS, we open-coded their responses to free-form questions about improvements to CS1, their experiences with the course lectures and labs, and their other comments.

We found that non-CS majors commonly expressed concerns about the difficulty of CS1 (24% of all respondents), the pace of the course (17%), and the high workload required of students (14%). Many of these students found it difficult to balance CS1's workload with their other courses, e.g. "The amount of information is overwhelming for people who had never programmed before. The course load is way [too] much, I personally have 4 other courses and it's taking up way too much time."

10% of all respondents specifically mentioned that they did not feel that CS1 was designed for non-majors or students without prior programming experience. Two of these students explicitly expressed a desire for an introductory programming course better suited to them: "Perhaps if there was a class that taught non-CPSC students how to program in Python and how that would be relevant for any degree [choice], it would be more attractive and get people hooked more"; "I wish there was a course that taught this at a slower and more applicable pace for non-cpsc students."

2.3 Summary

Our evaluations confirmed that students in degree programs with fewer CS majors failed the course at much higher rates. The surveys further confirmed that non-majors were less satisfied than majors with their overall experience in CS1, their expected grade, the workload, and what they learned in CS1. Non-majors also had less expert-like attitudes towards computer science. Not all of these differences are surprising. We would expect, for example, that CS majors might be more interested in computing than non-majors. However, being less interested in computing should not lead to a less satisfactory experience with the introductory course, nor does it mean that these students should be more fixed in their mindsets or be less able to appreciate the connections between computing and their everyday lives.

3 CS0.5 CURRICULUM DESIGN

The findings from the surveys, combined with CS1 instructor reflections and consideration of similar courses elsewhere, led us to a number of goals for CS0.5. We wanted the course to be relevant and engaging to students with various interests, follow the core curricular approach of CS1, have a flexible schedule and slower pace, foster positive attitudes about CS, be scalable to at least 500 students per term, and encourage students to reflect on their learning.

One approach to making CS courses relevant and engaging to non-CS students has been context-based CS1 courses (e.g. [1, 6, 10, 11, 13, 16]). However, rather than make multiple context-specific courses for students in different fields, we wanted to create a single, scalable course that provides flexibility in application topic area. Another common approach has been to use data analysis as a driving problem to introduce computer science concepts, programming, and data visualization (e.g. [2, 3, 18]). We wanted to maintain the focus on systematic program design from our CS1 course, while integrating the ideas of data analysis and visualization into CS0.5. Therefore, our main learning goal for CS0.5 is for students to complete a data analysis and visualization project in which they use computation to systematically solve a problem from a discipline of their choice. We centered the course design around a project because projects have been found to increase student engagement [12]. Every content module of the course focuses on the systematic program design skills and concepts that students will need to use in the project. Each student chooses a topic of interest for their project; we hoped that the ability to personalize the project would improve students' attitudes about computer science, especially students' personal interest in computing and the applications of computing to the real world.

To address concerns about workload and pace we reduced the overall number of learning goals in CS0.5 compared to CS1, and reduced the overall credit count compared to CS1 (3-credits vs. 4-credits). The CS0.5 students learn enough systematic program design to design a readable, well-structured, and well-tested program that uses Python's CSV module to read a .csv file, defines appropriate data types, analyzes the data to answer a specific question, and produces a graph to visualize the results. However, they do not tackle the latter 60% of the CS1 learning outcomes which cover more advanced topics (e.g. trees and generative recursion).

We chose to offer CS0.5 as a blended, reduced-face-time course, as we hoped that increased flexibility would lead to more accessibility for non-majors. Although the workload was the same as it would be for a non-blended course of the same credit count, the students have more control over when and where they complete their coursework. They attend one mandatory 80-minute lecture and one mandatory 50-minute tutorial per week (compared to two 80-minute lectures or three 50-minute lectures and one mandatory

3-hour lab in CS1). In each content module students are assigned pre-class work consisting of reading custom course materials, solving problems using the new concepts, and writing and submitting a brief summary of what they learned and at least one question about the module or how it relates to the rest of the course. In lecture we use Just-In-Time teaching [15] and deliver short mini-lectures to address the questions raised by students in their pre-class work. Students spend the majority of lecture time working in small groups to solve problems on a worksheet package; we usually interject with one or two 5-10 minute mini-lectures and/or wrap-up discussion that focuses on key problems from the worksheets. Completed worksheet packages are collected in the following lecture, graded for completion, and returned electronically. In tutorials, students work in small groups on weekly homework assignments with TA support. They also complete weekly peer review assignments that require them to solve a problem, assess three of their peers' solutions, and complete a self-assessment of their own solution.

We encouraged group work so that the students would have peer support if they started to feel discouraged. We also discussed the growth mindset [8] in class to foster positive attitudes, although we did not teach mindsets explicitly, as done by Cutts et al. [5]. To encourage students to reflect on their own learning, we included reflective questions on each worksheet package. For example, in one package we asked them "Which parts of the course are you finding the most difficult?" and "What is your strategy for mastering the parts of the course that you are finding the most difficult?".

4 CS0.5 YEAR ONE EVALUATION

We offered CS 0.5 for the first time in 2016/2017 Term 1 to 101 students (51% female), and then in Term 2 to 110 students (56% female); CS1 was offered in parallel to 1369 undergraduate students (39% female). We marketed CS1 to students intending to major in CS, as well as non-major students who felt it matched their needs and interests. We marketed CS0.5 as an alternative for non-majors who were interested in an introductory programming course that would allow them to solve problems in the discipline of their choice.

In both terms in 2016/2017 we conducted an evaluation of CS0.5 and CS1 to examine whether CS0.5 was successful in improving outcomes–measured in terms of pass rates, satisfaction, and attitudes–relative to the non-CS majors who remained in CS1. We conducted end-of-term surveys, and in CS0.5 we also ran Appreciative Inquiry focus groups to gain deeper insight into the students' experiences in the course and identify what they most valued about its design.

4.1 Proportions passing CS0.5 and CS1

The pass rates for each degree program are shown in Table 1. Overall, 83% of all students passed CS0.5 (n = 211) and 17% withdrew or failed. This contrasts with CS1 (n=1369), where 75% passed and 25% withdrew or failed. We saw notable increases in CS0.5 pass rates for all degree programs compared to CS1: the proportion was 7% higher for Science students, 15% higher for Arts students, 16% higher for Commerce students, and 12% higher for Other students. The outcomes in CS1 were fairly consistent with the previous year, differing by 5% or less for most groups. This suggests to us that the factors that influence different groups' success in CS1 are fairly steady year over year. A notable exception was the 13% increase in the proportion of Other students passing CS1, which may have been caused by more students in Other programs who expected to struggle in CS1 preferentially choosing CS0.5 instead.

Table 1: Proportion passing in CS0.5 and CS1 in 2016/2017.

	Science	Arts	Commerce	Other
CS0.5	88%	78%	93%	96%
CS1	81%	63%	77%	84%

4.2 Survey feedback in CS0.5 and CS1

In CS0.5 132 (66%) of the 201 students still registered at the end of term completed the survey, while in CS1, 726 (56%) of the 1307 students still registered completed it. We again asked respondents in the CS1 survey to declare their intended major(s): those students who chose computer science as at least one of their majors were classified as CS majors (52%), while all other respondents were classified as non-CS majors (48%).

We asked students in both courses the same set of satisfaction and CAS [7] questions as in the previous year's CS1 survey. We used one-way between-subjects ANOVAs to examine differences in satisfaction and attitudes between CS0.5 students and CS and non-CS majors in CS1. We performed post-hoc pairwise comparisons using the conservative Bonferroni correction. See Figures 3 and 4; all questions were optional, so the n varies for each group.

Satisfaction: Students' satisfaction varied significantly for overall experience, the expected grade, the amount of time and effort put into the course, and the skills and knowledge learned.

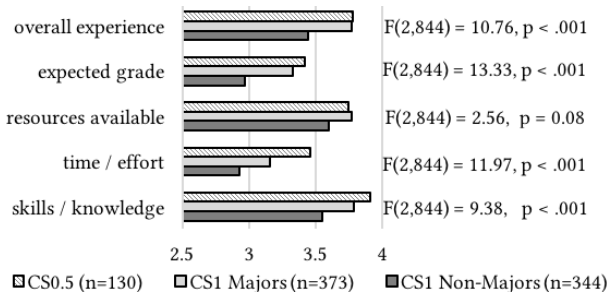

Figure 3: Between subjects ANOVA of satisfaction in 2017/2018; 1=very unsatisfied; 5=very satisfied.

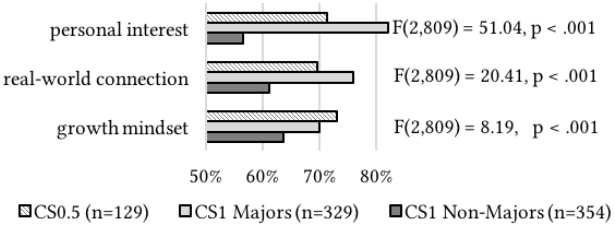

Figure 4: Between subjects ANOVA of students' agreement with expert opinions on CAS in 2017/2018.

Post-hoc comparisons showed that the satisfaction of non-CS majors in CS1 was significantly less than both CS majors in CS1 and CS0.5 students in their overall experience in the course ($p < .001$ and $p < .01$ respectively), their expected grade (both $p < .001$), the amount of time and effort ($p < .05$ and $p < .001$ respectively), and the skills and knowledge ($p < .01$ and $p < .001$ respectively). The satisfaction with the time and effort required was also less for CS majors in CS1 than students in CS0.5; there were no other significant differences between these two groups.

Attitudes towards computing: We found significantly different levels of expert agreement on the personal interest, real world connections and growth mindset subscales.

The post-hoc comparisons showed that non-CS majors in CS1 scored significantly lower than both CS majors in CS1 and CS0.5 students on the personal interest (both $p < .001$), real world connections ($p < .001$ and $p < .05$ respectively), and growth mindset (both $p < .01$) subscales. CS majors scored significantly higher on the personal interest ($p < .01$) measure than CS0.5 students; we found no differences between these groups on the other factors.

CS0.5 student comments: To explore the extent to which the observed outcome improvements could be explained by our curriculum design decisions, we analyzed students' responses to free-form questions about suggestions for improvement, their experiences with the lectures, tutorials, and project, and additional comments.

69% of all respondents remarked on how the project allowed them to apply skills in the course to their personal interests, e.g. "The project made me feel like everything I had learned was put towards a very interesting end-goal. It allowed me to understand the greater contexts within and beyond computer science and how it can be applied in different areas of life".

Students' comments also suggest that the course design supported them in developing a growth mindset in computing, with 33% of all respondents describing growth mindset related concepts or experiences. For example, some describe improving their skills tackling difficult problems: "I improved my ability to systematically work through a problem, looking at each specific aspect instead of immediately trying to go for the big picture, as that can be overwhelming." Others commented on the positive feelings associated with overcoming challenges. For example, one student wrote "the feeling of accomplishment whenever I figured out how to solve a problem is amazing. Anytime I mastered a new skill in general was great." Also related were students' comments on the benefit of having multiple opportunities to interact with course concepts and recognize that they needed help over time, e.g. "I love that ... we had a lot of chances to learn the material and ask for help. [If] we didn't understand it during pre-reading we could understand it during class, and then during the peer review, and then during the tutorial, and then during the practice problems."

4.3 CS0.5 student focus group feedback

In addition to comparing non-CS majors experience in CS0.5 with CS1 through the surveys, we also wanted to deeply explore the students' learning experience in CS0.5, identify the elements of the course design that course participants most valued, and solicit participant-driven ideas for improvements. To answer these questions, we conducted focus groups in CS0.5 using Appreciative Inquiry (AI) [19], an action-driven, participatory research methodology that focuses on what's working well and drives change by enhancing current strengths.

Focus group design: We conducted three 3-hour sessions with a total of 23 students and five teaching assistants. We conducted the first two sessions in the month following the first offering, and the third session the month following the second offering. Two facilitators led each session.

AI uses a four-phase process centered on an affirmative topic, in our case "CS0.5 at its best." In each 3-hour session, a different group of participants completed the first three phases of this process. In the first phase (*Discovery*), the participants conducted storytelling-based interviews in small groups to solicit stories that illustrated CS0.5 at its best. Each storyteller also identified what they valued about themselves or others in their story. The small groups then identified the three to five most salient themes from these stories and wrote them on a large piece of paper to share with the whole group via a gallery walk. Each participant voted on their top priority themes and we carried the most highly-rated themes into the next phase. The participants then formed new small groups, based on the priority theme that they were interested in, and collaboratively generated a drawing and a present-tense, one-sentence description (a provocative proposition) of how they envisioned CS0.5 at its best in relation to their theme (*Dream* phase). Finally, the participants brainstormed course design ideas that could enable this ideal future state (*Design* phase). The final AI phase, *Delivery* on changes, is too large to conduct in a 3-hour focus group; instead, course staff are using the findings to inform future course revisions.

Findings: We found several common themes that participants valued. Many of these aligned with our course design intentions, while some gave us new insight into non-majors' perspectives on their experiences learning CS that we intend to bring forward as we continue to iterate on the course. The project came up frequently, with groups using themes of 'versatility', 'flexibility', 'universal appeal' and 'personal connection' to describe the project and their appreciation of being able to apply new knowledge to their own interests. Envisioning what this could look like, one group wrote the provocative proposition that CS0.5 "enables students to incorporate their own interests and teaches skills that are relevant to their personal goals". Many felt that the project already does this successfully, but the design phase generated several suggestions for further improvement, e.g. collaborating with lab courses to use real observational data, grouping sections into disciplinary clusters, or providing students with more ways to showcase their final projects.

Several themes addressed the 'availability' (two groups), 'accessibility', and 'diversity' of resources, which the respondents appreciated because they could get the type of help they needed when they wanted it. One group's proposition described CS0.5 as "students actively learning how to think systematically by coding whenever they want, wherever they want, whatever they want." Design brainstorming in this area generated many new ideas for resources, such as weekly Q&A live streams, online chats for students who had trouble attending office hours, and dedicated CS0.5 study spaces to work with their peers outside of class time.

The growth mindset arose through several common themes related to the challenges that students encountered with the course concepts and assignments, and how they perceived these challenges:

'perseverance' (three groups) repeatedly came up as a quality that participants valued in themselves at their best in the course, a fourth group raised the related quality of 'self-reliance', and two groups cited the challenge and benefit of 'independent learning.' One group's proposition focused on students being "motivated by success to use resources and pursue solutions to challenging problems." Complementary to this focus on the individual were themes focused on the benefits of interactions with others: 'teamwork', 'support system', 'community' and 'peer review.' Along these lines, one group's proposition described "students asking each other questions, giving feedback, having discussions, and explaining their work," while another described "easily accessible, virtual and in-person resources to facilitate independent and collaborative student learning." These themes prompted many suggestions to enable even more avenues to support students and encourage them when things get tough, such as more opportunities for peer feedback, actively helping students get to know each other better, and new resources like the online chats or dedicated meeting rooms mentioned earlier.

5 DISCUSSION AND CONCLUSIONS

Our evaluation of the first two semesters of CS0.5 confirms that our course design decisions were successful in improving students' outcomes in CS0.5 relative to non-CS major students in CS1. The pass rates of students from all programs improved considerably in CS0.5 over CS1. Not only did students' satisfaction and attitudes improve in CS0.5 when compared with non-CS majors in CS1, we were encouraged to find that these were often in line with those of the CS-majors in CS1.

Survey respondents' reported satisfaction and comments suggest that CS0.5 students found the workload to be much more manageable than non-CS majors in CS1. Compared to non-CS majors in CS1, CS0.5 students were more satisfied not only with the time and effort that they expected to put into the course, but the grade they were able to earn with that effort, and in their overall experience. In addition, CS0.5 students were more satisfied with the skills they learned than non-CS majors in CS1. This suggests that the reduction in the number of learning goals made the course more manageable for non-CS majors, without hurting the course's perceived substance or value. Related to workload, the flexibility of the work-anywhere approach afforded by the blended classroom was identified as particularly valuable by several AI groups.

CS0.5 students' attitudes towards computing also improved on all measures. Compared to non-CS majors in CS1 we saw more expert-like growth mindsets, not only in the CAS scores, but also in the AI and survey comments made by CS0.5 students about overcoming challenges. The CS0.5 students also showed an increased interest in Computer Science and ability to connect computing to real world applications. The AI and survey comments revealed that the personalized project was particularly effective at promoting these attitudes and fostering positive experiences.

One limitation of this work is that the students who participated in evaluations may have been more engaged and thus some measures may be positively overestimated. This is especially likely in CS1, where instructors have noticed that many students who fail the course simply disengage and stop attending. This possibility is further suggested since survey respondents' pass rates are higher than for the courses overall (by 11% in 2015/2016 and 10% in 2016/2017 in CS1; by 8% in C0.5).

Appreciative Inquiry is not yet commonly used in CS education, but we hope our experience will encourage others to try using AI. The students provided rich, actionable feedback and we were impressed with their deep reflections on their learning experiences.

We plan to continue to evaluate the course via surveys and Appreciative Inquiry focus groups, and to respond to student feedback via curricular and pedagogical changes. We are happy to share our course materials; please contact the second author.

ACKNOWLEDGMENTS

We gratefully acknowledge the financial support provided by the UBC Carl Wieman Science Education Initiative and by UBC Vancouver students via the Teaching and Learning Enhancement Fund. We thank Steven A. Wolfman, Albert Xing, Alfred Xing, Giulia Mattia, and Erika Thompson for their contributions to this work.

REFERENCES

[1] Joel C. Adams and Randall J. Pruim. 2012. Computing for STEM Majors: Enhancing Non CS Majors' Computing Skills. In *Proc. SIGCSE '12*. 457–462. doi.org/10.1145/2157136.2157270
[2] Ruth E. Anderson, Michael D. Ernst, Robert Ordóñez, Paul Pham, and Ben Tribelhorn. 2015. A Data Programming CS1 Course. In *Proc. SIGCSE '15*. 150–155. doi.org/10.1145/2676723.2677309
[3] Austin Cory Bart, Ryan Whitcomb, Dennis Kafura, Clifford A. Shaffer, and Eli Tilevich. 2017. Computing with CORGIS: Diverse, Real-world Datasets for Introductory Computing. In *Proc. SIGCSE '17*. 57–62. doi.org/10.1145/3017680.3017708
[4] Tracy Camp, W. Richards Adrion, Betsy Bizot, Susan Davidson, Mary Hall, Susanne Hambrusch, Ellen Walker, and Stuart Zweben. 2017. Generation CS: The Growth of Computer Science. *ACM Inroads* 8, 2 (May 2017), 44–50. doi.org/10.1145/3084362
[5] Quintin Cutts, Emily Cutts, Stephen Draper, Patrick O'Donnell, and Peter Saffrey. 2010. Manipulating Mindset to Positively Influence Introductory Programming Performance. In *Proc. SIGCSE '10*. 431–435. doi.acm.org/10.1145/1734263.1734409
[6] Zachary Dodds, Ran Libeskind-Hadas, and Eliot Bush. 2012. Bio1 As CS1: Evaluating a Crossdisciplinary CS Context. In *Proc. ITiCSE '12*. 268–272. doi.org/10.1145/2325296.2325360
[7] Brian Dorn and Allison Elliott Tew. 2015. Empirical validation and application of the computing attitudes survey. *Computer Science Education* 25, 1 (2015), 1–36. https://doi.org/10.1080/08993408.2015.1014142
[8] Carol S. Dweck. 2006. *Mindset: The New Psychology of Success*. Random House.
[9] Matthias Felleisen, Robert Bruce Findler, Mathhew Flatt, and Shriram Krishnamurthi. 2001. *How to Design Programs*. MIT Press.
[10] Christian T. Jacobs, Gerard J. Gorman, Huw E. Rees, and Lorraine E. Craig. 2016. Experiences with efficient methodologies for teaching computer programming to geoscientists. *J. of Geoscience Ed.* 64, 3 (2016), 183–198. doi.org/10.5408/15-101.1
[11] Cynthia Bailey Lee. 2013. Experience Report: CS1 in MATLAB for Non-majors, with Media Computation and Peer Instruction. In *Proc. SIGCSE '13*. 35–40. doi.org/10.1145/2445196.2445214
[12] Patricia A. Marsh. 2007. What is known about student learning outcomes and how does it relate to the scholarship of teaching and learning. *Int. J. for the Scholarship of Teaching and Learning* 1, 2 (2007), 1–13. doi.org/10.20429/ijsotl.2007.010222
[13] Bruce A. Maxwell and Stephanie R. Taylor. 2017. Comparing Outcomes Across Different Contexts in CS1. In *Proc. SIGCSE '17*. 399–403. doi.org/10.1145/3017680.3017757
[14] Dawn C. Meredith and Edward F. Redish. 2013. Reinventing physics for life-sciences majors. *Physics Today* 66, 7 (2013), 38–43. doi.org/10.1063/PT.3.2046
[15] Gregor M. Novak. 2011. Just-in-time teaching. *New Directions for Teaching and Learning* 2011, 128 (Dec 2011), 63–73. doi.org/10.1002/tl.469
[16] Lauren Rich, Heather Perry, and Mark Guzdial. 2004. A CS1 Course Designed to Address Interests of Women. In *Proc. SIGCSE '04*. 190–194. doi.org/10.1145/971300.971370
[17] Linda J. Sax, Kathleen J. Lehman, and Christina Zavala. 2017. Examining the Enrollment Growth: Non-CS Majors in CS1 Courses. In *Proc. SIGCSE '17*. 513–518. doi.org/10.1145/3017680.3017781
[18] David G. Sullivan. 2013. A Data-Centric Introduction to Computer Science for Non-Majors. In *Proc. SIGCSE '13*. 71–76. doi.org/10.1145/2445196.2445222
[19] Diana Whitney and David Cooperrider. 2011. *Appreciative inquiry: a positive revolution in change*. ReadHowYouWant. com.