A Project-Based Curriculum for Computer Science Situated to Serve Underrepresented Populations

Rebecca Bates
Jonathan Hardwick
bates@mnsu.edu
jonathan.hardwick@mnsu.edu
Minnesota State University, Mankato
Mankato, MN, USA

Guarionex Salivia
Lin Chase
guarionex.salivia@mnsu.edu
lin.chase@mnsu.edu
Minnesota State University, Mankato
Mankato, MN, USA

ABSTRACT

Building on known best practices, this paper describes a newly implemented project-based curriculum for undergraduate computer science majors. The program, designed to provide all students the known benefits of the project-based approach, is being implemented in a context that focuses heavily on drawing students of color, women, and other underrepresented groups into STEM careers. Our Fall 2021 upper-division entering student cohort is 70% female and 80% people of color. Our new computer science program builds on our strong and internationally recognized track record in applying project-based learning (PBL) in engineering. We believe our PBL CS program to be the first program of its type in North America. Key components of the program are: project-based learning across multiple courses in the upper-division, project-based learning within lower-division courses, straightforward articulation with regional and national community colleges that are minority serving institutions (MSIs), industry and research connections that drive projects, and vertical integration of teams across upper-division semesters. Core to this is a commitment to addressing access and equity issues. The foundation for the new upper-division program is a revamped lower-division core curriculum that also feeds into the information technology and information systems majors offered at our school. The core curriculum is designed to reduce barriers to a degree in computing while encouraging more students to explore a degree in the math-intensive computer science program. This curricula initiative paper describes the curriculum, the supporting theory, implementation challenges, and current outcomes.

CCS CONCEPTS

• **Social and professional topics → Computer science education**; **Model curricula**.

KEYWORDS

Computing Education, Project-based Learning, Computer Science Curriculum, Diversity in Computing, Inclusion

ACM Reference Format:
Rebecca Bates, Jonathan Hardwick, Guarionex Salivia, and Lin Chase. 2022. A Project-Based Curriculum for Computer Science Situated to Serve Underrepresented Populations. In *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education V. 1 (SIGCSE 2022), March 3–5, 2022, Providence, RI, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3478431.3499312

1 INTRODUCTION & MOTIVATION

Public education faced significant funding challenges during the US recession of 2008. In response, the computer science program at our university was eliminated, while programs in information technology and systems were retained. In the intervening years, the university has made significant investments in creating effective and popular project-based engineering programs [3, 6], and in the process has developed an international leadership role in project-based learning (as distinct from problem-based learning where problems may motivate learning but are typically at a smaller scale and are addressed within individual courses). Our wish was to apply this successful approach to rebuild the computer science program. Thus we went through several iterations of cross-disciplinary and cross-organizational planning in order to:

- Apply project-based learning on an exclusive basis in the upper division, based on the existing successful project-based engineering programs
- Create a program that makes participation accessible and workable for students from underrepresented populations in the STEM fields
- Connect both national and regional industry participants as project partners and career accelerators for our students
- Prepare our local students for this project-based approach during lower-division courses, including creating a common lower-division core curriculum to be shared by all programs requiring computing classes
- Coordinate carefully with the regional and national community colleges that serve as student pathways to our program
- Focus on student inclusion and student access to degree pathways at all levels
- Provide training for faculty who will be shifting to development and teaching practices used in project-based learning

During the planning process, we referred regularly to the literature in project-based learning and related topics so that participating faculty and other colleagues unfamiliar with the project-based learning approach could gain a deeper understanding. We also looked closely at existing undergraduate project-based programs in computer science for inspiration and guidance. As far as we know, ours is the first such program in North America.

The result of our multi-stage planning process is an articulation of program values and outcomes. We have also now launched our first full cohort of upper-division students.

In this curricula initiative paper we provide our literature- and experience-inspired program values, a brief description of the motivation for project-based learning and examples of the benefits seen in other programs, and the development of the common lower-division core and the upper-division project-based program. Discussion includes how we address inclusion, assessment, and implementation challenges, concluding with initial assessment data from the first year of upper-division projects, statistics on our current student population, and our next steps for bringing this approach to computer science education to student learners.

2 PROGRAM VALUES

Our planning process resulted in a clear statement about the philosophy of the major and a list of values that ground the major, both in its development and implementation. Having these as an agreed point of reference for all parties was a tremendous help during the multi-stage planning process. The university's new computer science major is a project-based program where students choose to attend the institution because of the program. By providing a different learning experience in the field of computer science, and by situating the program in a well-designed set of feeder pathways into the program from an array of minority serving institutions (MSIs), this program attracts and retains a student body intended to increase diversity in the field. Admission focuses on including students who thrive in a community-driven, project-based approach.

Our program encourages diversity of experience, ideas, and background in admissions, projects, and learning. Students are prepared to enter industry or graduate school and to work in diverse, inclusive environments. Students have experiences with the breadth and depth of computer science theory as well as significant, successful experiences applying theory in multiple contexts.

The computer science major is grounded in the following values:

(1) Significant, meaningful projects and research experiences give context to student learning at multiple points in the program [5].
(2) Students and faculty have autonomy [8, 27] in both content and learning approaches.
(3) Self-directed and self-motivated learning [9] is encouraged and scaffolded. Students completing the program are able to continue to learn and develop in areas that will support their careers and personal interests.
(4) Student engagement is encouraged and supported at all levels in the program, supported by meaningful projects, autonomy and self-directed learning [4, 22].
(5) A cohesive four-year experience includes multiple entry points up to the junior year for transfer students.
(6) A community of learners [26] is created to allow students to work in teams in multiple contexts, both formal and informal.
(7) An inclusive environment [22] that provides diverse experiences encourages recruitment and retention of underrepresented students and will assist over-represented students in developing skills to work and thrive in inclusive work environments.

These values continuously inform the curriculum development and faculty training processes.

3 PROGRAM OUTCOMES

We considered the likelihood of future accreditation. Defining program outcomes allows for an assessment process to be built alongside the curriculum. Beginning with the ABET-defined six program outcomes that describe what computer science students should be able to do at graduation [1], our program adds a seventh:

(1) Analyze a complex computing problem and to apply principles of computing and other relevant disciplines to identify solutions.
(2) Design, implement, and evaluate a computing-based solution to meet a given set of computing requirements in the context of computer science.
(3) Communicate effectively in a variety of professional contexts.
(4) Recognize professional responsibilities and make informed judgments in computing practice based on legal and ethical principles.
(5) Function effectively as a member or leader of a team engaged in activities appropriate to computer science.
(6) Apply computer science theory and software development fundamentals to produce computing-based solutions.
(7) Work successfully in a diverse, inclusive environment.

For a program that will initially house a small number of students (i.e., ~24-40 upper-division students in the first few years), program assessment can be portfolio-based and focus on upper-division success with feedback provided for lower-division courses.

4 PROJECT-BASED LEARNING

Project-based learning (PBL) as a tool for computer science education is touted for K-12 learners (e.g., [15, 16, 30]), but less used as an overarching approach in undergraduate curricula. Projects are frequently embedded within classes, and focus on the development of contextual skills and knowledge [11, 13, 19, 25]. This approach has been used to address retention issues in lower-division courses (e.g., [31]). Capstone projects are also used to finalize undergraduate learning (e.g., [10, 14, 23, 24]) but many professional skills remain underdeveloped. Aalborg University uses overarching projects in all fields and serves as an example for the approaches at our university [20, 21]. Programs like Arizona State University's software engineering degree address professional skills alongside software design at varying levels for each year [13], so extending this approach to computer science is reasonable.

The distinguishing factor of this curriculum initiative is that projects, while embedded within courses in the lower division, are used as the focus of a semester's learning in the upper division. Supported by smaller core and elective classes, student teams apply their knowledge in meaningful ways while developing computer science skills in analysis, design, and evaluation as well as professional skills like teamwork, time management, ethical judgement, and communication.

Benefits of project-based learning include potential for recruitment of non-traditional students, retention of students, and engaged learning. These benefits have been shown within our engineering programs [6, 28, 29]. The computer science program builds on components of our engineering programs, that include 1) overarching design projects in each semester of the upper-division program that require knowledge from multiple technical topics, 2) technical competency learning that first addresses breadth in single credit units and allows for deeper learning for all students in a broad range of electives, some of which are student-led (and faculty-evaluated), and 3) student autonomy as shown through selection of projects, electives, and learning approaches.

Much prior work builds the case for PBL in engineering. There are obvious differences between computer science and engineering, however, we hypothesize that PBL will still be valuable in this context. We believe this project-based approach is worth implementing so that professional skills are developed in the context of computational learning and the meaning and value of that learning is reinforced through external need (often shown by industry partners). The extension to computer science includes exploratory projects (research) and addressing canonical problems like satisfiability, complexity, or optimization problems.

5 LOWER DIVISION APPROACH

We first focused on developing learning objectives and designing a lower-division common core that serves three undergraduate major programs. Though not typically regarded as an interdisciplinary degree, it is worth noting that computer science benefits and is benefited by a plethora of disciplines. Thus, the design took into consideration that the common core would need to serve all the programs in our institution that relate in some way to the computing disciplines. Looking at guidance from the accrediting body ABET [1], and at curricula recommendations from the Association for Computing Machinery (ACM) [2] for the first two years of a computer science program, we developed a mapping of courses that would constitute a basis for our existing degrees of management information systems, information technology, computer engineering, health informatics and cognitive science, as well as the new computer science degree. It would also allow for other degrees to opt into this set of courses, and would help future development of curricula around this common set of courses.

The lower-division common core refers to the intentional design of a cluster of courses that satisfies the common learning objectives associated with the first two years of the computing, with areas of study either building off the core or intersecting with it, as shown in Figure 1. This common core should satisfy the following objectives:

- to meet the needs of computer science majors,
- to provide students with the basis of computing needed for multiple disciplines,
- to enable students' mobility across majors,
- to simplify the mapping of lower level computing courses to ACM and/or ABET learning objectives,
- to allow for ease of update and assessment of learning objective of lower level computing courses,
- to reduce duplication of computing courses in curricula.

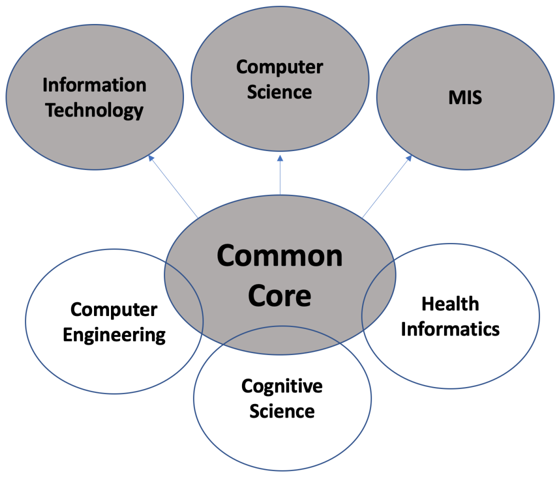

Figure 1: Common Core Curriculum Design

Table 1: Common Core Courses

Introduction to Programming	Data Structures
Algorithms	Computer Architecture
English Composition	Calculus 1

We also planned aggressively for computer science transfer pathways in order to allow smooth articulation between community college graduates and the upper-division program. This was especially important because our success in recruiting and retaining students from underrepresented populations is highly dependent on building and maintaining pathways from the minority serving institutions in our network.

The six courses that comprise the lower-division common core are shown in Table 1. Students intending to major in to computer science or information technology take all of these courses, while the curricula for management information systems, health informatics, computer engineering, and cognitive science intersect this set of courses. Students in any of these areas can therefore concentrate their effort on completing this common core, and later decide which area of study to pursue. Alternatively, students in later stages of their academic paths, may decide to change majors and the common core facilitates their movement across the majors that take advantage of the common core design.

The recommended first two years for computer science majors is shown in Table 2 and includes the foundational courses for a minor in mathematics. Together with computing courses that needed to be created and/or revised, there was also a need to create a new course which we call "Discrete Mathematics for Computer Science," a course that complements the common core, prepares students for more complex proofs, and that computer science students take in parallel with the Algorithms course.

Project-based learning at the course level is introduced in lower-division courses such as Data Structures, Algorithms and Computer Architecture. Students are not only introduced to the theoretical concepts according to these courses' objectives, but work on small

Table 2: First two years of 4-year plan towards CS degree, including lower-division common core (noted with *)

First Year

Fall	Spring
Intro to Programming*	Data Structures*
Calculus 1*	Calculus 2
English Composition*	Science Elective
General Education course	General Education course

Second Year

Fall	Spring
Algorithms*	Computer Architecture*
Discrete Mathematics for CS	Linear Algebra
Science Elective	Communication Studies course
General Education course	General Education course

projects using technologies such as Arduino and Raspberry Pi. These small projects serve to support the program values of providing meaningful project experiences and helping students become self-directed and self-motivated via scaffolding.

6 UPPER-DIVISION APPROACH

Application to the upper-division computer science program typically takes place at the end of a student's second year. Admission is not guaranteed, as space is limited. Many students will continue on from the common core lower-division courses into computer engineering, information technology, and management information systems programs, and we expect that only a minority will apply to computer science. However, by potentially delaying this decision point until the end of their second year, students have the freedom to transfer between intended pathways in their first two years with minimal friction and cost to themselves.

We believe that students who learn well when there are hands-on and direct experiences implementing the required theory will thrive in our upper division, as shown by the experiences of students in our project-based engineering programs. These students will develop their professional skills by working in a deadline-driven, team-based environment, that still allows flexibility in their learning [17, 18]. Applicants are therefore asked to submit two reflective essays on why they want to participate in a project-based program and what they will contribute to the community of learners, along with a code sample. These essays count alongside demonstrated performance in lower-division courses in the application process. In addition, a faculty recommendation or advocate will let the admissions committee take into account student qualities that might not be evident on a transcript, particularly those that will benefit the inclusive learning community and project teams.

Project-based learning is implemented in the upper division via semester-long team projects, with faculty acting as mentors and advisors to teams of students who learn computer science in the process of solving different client or customer problems. Students undertake four of these projects during their final two years, typically as part of a different team each time. Projects come from industry, faculty research, other areas of the university, or canonical problems, but they must have a well-defined customer (or faculty advocate of a canonical problem) who can engage with the students during the process. Most projects will use an Agile project-management approach, where frequent customer contact is an integral part of the process. Past and future projects include:

- Updating a medical calculator smartphone app to follow new design guidelines, and to be cross-platform.
- Developing custom firmware for Linux embedded systems.
- Adding fuzz testing of new functionality in a domain-specific language.
- Creating a game to showcase new features in a commercial virtual reality platform.
- Building a cloud-based web application to replace the spreadsheet interface of a custom workflow system.
- Prototyping a machine-learning system to predict chord progressions based on a reference corpus of songs.

Teams typically consist of 3-5 students. Where possible these students come from different cohorts, as vertical integration of teams promotes transfer of knowledge and expectations between cohorts, and more closely matches the typical experience in industry. This allows students to grow in leadership experience as they mentor near-peers. Especially large or tricky customer projects can be spread out across more than one semester, with one team performing knowledge transfer to another, which is another important industry skill.

Credit for projects is earned through 4-credit project classes. While students regularly meet with faculty mentors, course hours are not classroom time, but are used for team interactions and presentations. The professional skills associated with outcomes 3 (communication), 4 (professional responsibility), 5 (teamwork), and 7 (inclusion) are addressed both in project work and during a weekly, one-credit seminar. Tools and approaches are presented in seminar, applied in the project, and then evaluated as part of the professionalism component of the project grade. Outcomes 1 (analysis), 2 (design, implement, evaluate), and 6 (apply theory) are evaluated via additional components of the project grade.

Alongside these projects, the first year of the upper division includes four required computer science core courses, covering the major topics of operating systems, software engineering & parallel computing, programming languages, and databases & information security. These core courses are not intended to cover the full breadth and depth of the course topics, as would be the case for a classical computer science curriculum. Instead, they are "half-height" 2-credit courses, designed to give students a broad overview of each topic and prepare them for further project-based investigation of individual aspects of those topics or further learning in elective courses. The topics of these courses were chosen to complete coverage of the core experiences of recommended ACM outcomes when combined with the lower-division core [2] (and we look forward to updating these with the new curricular guidelines [12]). Although one credit courses are used in our engineering programs to explore the breadth of a topic such as fluid mechanics or digital logic (a somewhat closer step towards competency-based assessment), this program uses 2-credit courses to better fit with faculty loads in a department with 4-credit courses. While we see it working well for electives, our core classes might challenge a

Table 3: Final two years of 4-year plan. Note that CS Core and CS Elective classes are 2 credits instead of 4

Third Year

Fall	Spring
Seminar (1 credit)	Seminar (1 credit)
CS Project 1	CS Project 2
CS Core: Operating Systems	CS Core: Prog. Languages
CS Core: Software Engineering & Parallel Computing	CS Core: Databases & Information Security
Discrete Maths for CS 2	Probability & Statistics
General Education course	CS Elective

Fourth Year

Fall	Spring
Seminar (1 credit)	Seminar (1 credit)
CS Capstone Project 1	CS Capstone Project 2
CS Elective	CS Elective
CS Elective	CS Elective
CS Elective	General Education course
General Education course	General Education course

single teacher (e.g., Software Engineering & Parallel Computing). We address this by having multiple faculty members collaborate to develop these courses.

The required core courses are followed by six 2-credit elective courses spread across the final three semesters. These elective courses provide space for faculty to develop tailored instruction to support and develop students as they explore increasing depth of computer science topics in their projects. Where possible, projects are chosen so that a single elective course can be used to support more than one team, reducing the teaching load on faculty. For example, given two student projects covering different customer problems in speech recognition and computer vision, a single elective providing an introduction to artificial intelligence techniques can provide both teams with a jump-off point for their research and learning. The elective courses also allow for significant student autonomy, allowing students to undertake individual study to further explore areas they are especially interested in while earning credit for doing so. The full upper-division program is shown in Table 3.

7 ADDRESSING INCLUSION

One of the most important aspects of this program's support for inclusion has already been touched on in previous sections. By specifically designing our program to recruit and retain students transferring from a wide network of MSIs, both in our state and beyond, we provide an accessible pathway for community college graduates to apply for and step into our upper-division program. Because the populations at these feeder institutions are highly diverse (the average is 50% representation of underrepresented populations), our curricular approach, coupled with active recruitment, mentoring, and other retention support, significantly expands access to the study of computer science.

Our lower-division common core is intentionally designed so that students can switch between degree pathways with minimal friction during their first two years. The combined classes during these first two years give students exposure to classmates on different pathways, affording them more opportunities to socially explore alternative futures for themselves.

A sense of belonging and strong learning communities have been important factors in the success of our university's project-based engineering programs, and we are replicating this success in the computer science program. A big part of this comes from the provision of student team rooms, dedicated to the program and to which the students always have access. Teams are assigned their own space for project work, and in our experience rapidly acquire a sense of shared ownership of this space. The seminar also promotes a sense of unity, by gathering the entire group into a single room each week for announcements, feedback, and content delivery to support project work. This strengthens social bonds and supports shared norms and expectations. Co-curricular sessions including industry speakers and faculty research discussions for all undergraduate computing students further creates connections between lower-division and upper-division students, making the pathways through the degrees more visible to first and second year students.

We are also promoting inclusion by encouraging students not already in a computing major to consider the program. To do this we have brought back an introductory CS 0 course, based on the AP Computer Science Principles curriculum [7]. The course is targeted specifically at students with no previous experience, and combines a broad overview of the computing field with some introductory programming and computational thinking. It fulfills a general education requirement at our university, and can therefore provide a pathway into the discipline for students who may not have been initially interested in anything other than checking off a requirement.

Students who are struggling in lower-division computing common core and math classes can also get help from dedicated tutors at our university and department tutoring centers. These tutors are typically seniors with existing experience of the program trained to support student learning, further promoting inclusion.

Along with the structural design to allow multiple pathways, we have considered our additional graduation outcome that students will have an ability to "work successfully in a diverse, inclusive environment." This outcome has been addressed in our engineering programs through training during seminars and active feedback about team interaction in professionalism grades. It is also crucial to develop a culture where non-inclusive behavior is addressed and corrected, with opportunities for all to grow and improve. This approach has transferred to the computer science program.

It is possible to seem hypocritical when addressing issues of professionalism and inclusion and it is just as important for faculty to be willing to grow, change, and improve as we expect students to be. Therefore, regular student feedback about the program and their experience is used to gauge the experiences of inclusion as well as to further include students in the ultimate success of the program. This bridges inclusion as a practice with assessment.

8 ASSESSMENT

Accrediting engineering and computer science programs requires a process of continuous improvement. There are multiple ways to gather the information necessary for this process. These are some of the approaches we have used in other programs and now use here.

We have not made any changes to student assessment techniques in the lower-division common core courses. Many students in these courses will be going on to programs other than computer science, and it would be unfair to impose any radical changes on their assessment. However, as previously noted we do encourage the use of project-based learning techniques *within* these courses. Our intent is that early exposure to small-scale project-based learning techniques will encourage students to develop their skills, regardless of their final program choice, and will help them decide whether they want to enter a project-based program. This can be evaluated through assessment of upper-division students.

Assessment in the upper division relies heavily on student portfolios. The program as a whole defines a wide range of "performance indicators" associated with program outcomes that students must demonstrate in order to graduate. A student's portfolio then consists of their completed courses and projects, plus all of the evidence for performance indicators that they have gathered. This provides students in the program with a clear idea of the competencies and knowledge that they are expected to achieve, and students can tailor their project and course experiences to progress through the performance indicators, repeating as necessary to fill in any weak spots in their portfolio. For example, a student who has already been a team member on a successful project might request to be project lead on their second project, with a specific goal of fulfilling their performance indicators in the leadership area. Even if the second project is also highly successful, the student may be judged to have not completed all of the leadership performance indicators, for example if there was significant dysfunction or poor management practices within the group. In this case, the student will be expected to continue to work on their leadership performance indicators in future projects, possibly after undertaking self-directed study in the area to better prepare themselves for their second attempt.

Assessment of project work culminates in individual oral final exams every semester, where students are expected to provide evidence and justify how they have met the expectations of the program. This evidence is also included in their portfolio, and students therefore graduate with a complete "interview-ready" portfolio which showcases their accomplishments and growth during the program. The performance indicators and sample student portfolios will form an integral part of future accreditation of the program, as they provide clear evidence of its intent and success.

Student feedback is another crucial part of assessment. Practicing the values of student-motivated learning and student engagement can be enhanced by listening to and incorporating student feedback. When all students are heard in this process it is another way to be inclusive of students, thus enacting the value of inclusion even in our assessment process. Formal feedback sessions are held at the end of each semester and informal feedback is encouraged.

Finally, faculty reflection on our perceptions of success and failure, provides additional opportunities for assessing the program and creating improvement plans. Regular faculty summits allow for institutionalization of this process and for sharing of information across course offerings and program faculty.

9 IMPLEMENTATION CHALLENGES

Rejuvenation of the computer science program, particularly as an exclusively project-based program, has met and will continue to meet with some challenges.

At our university, resource allocation depends greatly on student enrollment. Our project-based program creates demands for both faculty and space above the existing baseline. Our project-based learning approach requires a higher upper-division faculty to student ratio than the existing traditional classroom approach. Additionally, we dedicate significant classroom space to full time use for project teams. We are working hard with many different organizations within our university (including academic leadership, facilities, fundraising, and legal teams) and beyond (industry partners and independent sources of support) to create the cultural and financial shifts needed for a sustainable working model.

We also tried to be creative in designing a computer science program that would not suffer from historical trends of students migrating to less math-intensive majors, and a sense of competition between programs. Instead we have focused our efforts on integration, resulting in the implementation of the lower-division common core shared by all computing majors.

Another major challenge is the significant change in faculty expectations when teaching in a project-based learning environment. Students require much more feedback and guidance, especially at early stages, and faculty need to allow much more flexibility than with a standard teaching load. Faculty mentoring and frequent interventions are not typical expectations of faculty who are accustomed to regular lectures and examinations. Furthermore, faculty involved with the program need to find appropriate projects for students on a regular basis. While much of these are challenges that can be overcome with faculty training, it should be acknowledged that faculty time spent on project guidance cannot be spent on research or other forms of service.

10 CURRENT OUTCOMES AND NEXT STEPS

A small cohort of pioneering upper-division students started the program as first semester juniors in Fall of 2020, embarking on the sequence of four project semesters which will lead them to be the first graduates of the program in Spring of 2022. Also in Fall of 2020, the computer science curriculum was formally published in our bulletin. Since then students have been entering as first and second year students completing their lower-division core, on target for graduations starting in 2023. For Fall of 2021 we received a large number of transfer applicants for the upper-division entry. This cohort is drawn 50% from our own lower division and 50% from transfer students from within our state (40%) and other states and countries (60%). In line with our planning to especially serve underrepresented student populations, this new entering cohort is 70% women, and 80% people of color. We look forward to reporting on the progress of our students and program assessment data as well as our implementation successes and challenges in future years.

11 ACKNOWLEDGEMENTS

The authors gratefully acknowledge the leadership of deans supporting project-based programs in our college, the collegial cooperation of the CS Strike Force made up of faculty and staff from computer engineering, computer information science, and integrated engineering, and the work of department faculty in both learning new methods and presenting these to students.

REFERENCES

[1] ABET. 2020. Criteria for Accrediting Computing Programs, 2020-2021. https://www.abet.org/accreditation/accreditation-criteria/criteria-for-accrediting-computing-programs-2020-2021/

[2] ACM. 2013. Computer Science Curricula 2013. https://doi.org/10.1145/2534860

[3] Cheryl Allendoerfer, Rebecca A. Bates, Jennifer Karlin, Ronald R. Ulseth, and Dan Ewert. 2015. Leading Large-Scale Change in an Engineering Program. In *2015 ASEE Annual Conference & Exposition*. ASEE Conferences, Seattle, Washington. https://peer.asee.org/24397.

[4] K. Bain. 2012. *What the best college students do.* Harvard University Press, Cambridge, MA.

[5] R. Bass and H. Elmendorf. [n. d.]. Designing for difficulties: Social pedagogy as a framework for course design. ([n. d.]). https://blogs.commons.georgetown.edu/bassr/social-pedagogies/

[6] Rebecca Bates, Elizabeth Pluskwik, and Ron Ulseth. 2020. Startup of an Innovative Program x3 – Iron Range Engineering Propagated. In *2020 IEEE Frontiers in Education Conference (FIE)*. 1–4. https://doi.org/10.1109/FIE44824.2020.9274291

[7] College Board. 2020. AP Computer Science Principles: Course and Exam Description. https://apcentral.collegeboard.org/pdf/ap-computer-science-principles-course-and-exam-description.pdf

[8] D. Bruff. 2011. Social pedagogies: Authentic audiences and student motivation. (February 2011). https://derekbruff.org/?p=808

[9] E. L. Deci and R. Flaste. 1995. *Why we do what we do: The dynamics of personal autonomy.* Putnam's Sons, New York, NY.

[10] Robert F. Dugan Jr. 2011. A survey of computer science capstone course literature. *Computer Science Education* 21, 3 (2011), 201–267. https://doi.org/10.1080/08993408.2011.606118 arXiv:https://doi.org/10.1080/08993408.2011.606118

[11] S. Fincher and M. Petre. 1998. Project-based learning practices in computer science education. In *FIE '98. 28th Annual Frontiers in Education Conference. Moving from 'Teacher-Centered' to 'Learner-Centered' Education. Conference Proceedings (Cat. No.98CH36214)*, Vol. 3. IEEE, 1185–1191 vol.3.

[12] Stephen Frezza, Arnold Pears, Marisa Exter, and Barry Lunt. 2018. Crafting the Future of Computing Education in CC2020: A Workshop. In *Proc. 2018 ASEE Annual Conference & Exposition.*

[13] K. Gary. 2015. Project-Based Learning. *Computer* 48, 9 (2015), 98–100.

[14] Wouter Groeneveld, Joost Vennekens, and Kris Aerts. 2019. Software Engineering Education Beyond the Technical: A Systematic Literature Review. arXiv:cs.SE/1910.09865

[15] Ting-Chia Hsu, Shao-Chen Chang, and Yu-Ting Hung. 2018. How to learn and how to teach computational thinking: Suggestions based on a review of the literature. *Computers & Education* 126 (2018), 296 – 310. https://doi.org/10.1016/j.compedu.2018.07.004

[16] Peter Hubwieser, Michal Armoni, Michail N. Giannakos, and Roland T. Mittermeir. 2014. Perspectives and Visions of Computer Science Education in Primary and Secondary (K-12) Schools. *ACM Trans. Comput. Educ.* 14, 2, Article 7 (June 2014), 9 pages. https://doi.org/10.1145/2602482

[17] Bart Johnson and Ronald Ulseth. 2015. Professional Competency Development in a PBL Curriculum. In *Proceedings of the 5th International Research Symposium on PBL, part of International Joint Conference on the Learner in Engineering Education (IJCLEE 2015).*

[18] Bart Johnson, Ronald Ulseth, Crystal Smith, and Derek Fox. 2015. The impacts of project based learning on self-directed learning and professional skill attainment: A comparison of project based learning to traditional engineering education. *2015 IEEE Frontiers in Education Conference (FIE)* (2015), 1–5.

[19] Judy Kay, Michael Barg, Alan Fekete, Tony Greening, Owen Hollands, Jeffrey H. Kingston, and Kate Crawford. 2000. Problem-Based Learning for Foundation Computer Science Courses. *Computer Science Education* 10, 2 (2000), 109–128. https://doi.org/10.1076/0899-3408(200008)10:2;1-C;FT109 arXiv:https://doi.org/10.1076/0899-3408(200008)10:2;1-C;FT109

[20] Finn Kjersdam. 1994. Tomorrow'neering Education— The Aalborg Experiment. *European Journal of Engineering Education* 19, 2 (1994), 197–204. https://doi.org/10.1080/03043799408923285 arXiv:https://doi.org/10.1080/03043799408923285

[21] Anette Kolmos. 2017. *PBL CURRICULUM STRATEGIES: From Course Based PBL to a Systemic PBL Approach.* Brill | Sense, 1–12.

[22] G. D. Kuh. 2008. *High-impact educational practices: What they are, who has access to them and why they matter.* Association of American Colleges and Universities.

[23] John W. McManus and Philip J. Costello. 2019. Project Based Learning in Computer Science: A Student and Research Advisor's Perspective. *J. Comput. Sci. Coll.* 34, 3 (Jan. 2019), 38–46.

[24] J. J. Olarte, C. Domínguez, A. Jaime, and F. J. García-Izquierdo. 2016. Student and Staff Perceptions of Key Aspects of Computer Science Engineering Capstone Projects. *IEEE Transactions on Education* 59, 1 (2016), 45–51.

[25] Robert Pucher and Martin Lehner. 2011. Project Based Learning in Computer Science – A Review of More than 500 Projects. *Procedia - Social and Behavioral Sciences* 29 (2011), 1561 – 1566. https://doi.org/10.1016/j.sbspro.2011.11.398 The 2nd International Conference on Education and Educational Psychology 2011.

[26] R. M. Ryan and E. L. Deci. 2000. "Self-determination theory and the facilitation of intrinsic motivation, social development, and well-being". *"American Psychologist"* 55, 1 (2000), 68–78.

[27] P. Tough. 2016. *Helping children succeed: What works and why.* Houghton Mifflin Harcourt, Boston, MA.

[28] Ronald R. Ulseth. 2016. Development of PBL Students as Self-Directed Learners. In *2016 ASEE Annual Conference & Exposition.* ASEE Conferences, New Orleans, Louisiana. https://strategy.asee.org/26823.

[29] Ronald R. Ulseth and Bart Johnson. 2015. Iron Range Engineering PBL experience. In *Proceedings of the Seventh International Symposium on Project Approaches in Engineering Education (paee'2015), Integrated in the International Joint Conference on the Learner in Engineering Education (ijclee'2015) Event.*

[30] Nasanbayar Ulzii-Orshikh and John Dougherty. 2020. Iteration with Intention: Project-Based Learning of Computational Thinking. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education (SIGCSE '20).* Association for Computing Machinery, New York, NY, USA, 1289. https://doi.org/10.1145/3328778.3372651

[31] Petri Vesikivi, Minna Lakkala, Jaana Holvikivi, and Hanni Muukkonen. 2020. The impact of project-based learning curriculum on first-year retention, study experiences, and knowledge work competence. *Research Papers in Education* 35, 1 (2020), 64–81. https://doi.org/10.1080/02671522.2019.1677755 arXiv:https://doi.org/10.1080/02671522.2019.1677755