Infusing Computing Identity into Introductory Engineering Instruction

Briana Bettin*, Michelle Jarvie-Eggart†, Kelly S. Steelman‡ and Charles Wallace§
*Department of Computer Science
Michigan Technological University, Houghton, Michigan 49931-1295
Email: bcbettin@mtu.edu
†Department of Engineering Fundamentals
Michigan Technological University, Houghton, Michigan 49931-1295
Email: mejarvie@mtu.edu
‡Department of Cognitive and Learning Sciences
Michigan Technological University, Houghton, Michigan 49931-1295
Email: steelman@mtu.edu
§Department of Computer Science
Michigan Technological University, Houghton, Michigan 49931-1295
Email: wallace@mtu.edu

***Abstract*—In this work in progress/innovative practice paper, we describe our efforts to integrate introductory computer science pedagogical methods into an engineering fundamentals context. Our multidisciplinary team is addressing the problem of perceived value and applicability of programming for first-year engineering students. These students tend to be unaware of the importance of programming skills for practicing engineers within industry, and consequently display low interest in programming. We are working to address this perception problem through interventions in the engineering fundamentals classroom.**

Our interventions have three objectives: establish awareness of how programming skills can be generalized beyond the introductory classroom, incorporate targeted activities for algorithmic thinking, and demonstrate the practical applications of programming skills. Similar interventions are part of the introductory experience for our computer science students; our goal here is to tailor the interventions to an introductory engineering context. Our goal is for students to recognize computing as part of the career and identity of an engineer.

I. Casting Engineers as Coders

Within industry and across engineering disciplines, computing skills and tools have become essential for engineers. As the Industrial Internet of Things (IIoT) increasingly embeds robotics, software, and electronic controls within manufacturing processes, engineers need to be able to extract data, interpret, and write code to control these processes [1]. For example, one of our authors worked in industry as an environmental engineer, where she extracted emissions data from a process control system for federal reporting.

The Accreditation Board for Engineering and Technology's (ABET) Criteria for Accrediting Engineering Programs Criterion 5.b (Curriculum) states that engineering curricula must include "a minimum of 45 semester credit hours (or equivalent) of engineering topics appropriate to the program, consisting of engineering and computer sciences and engineering design, and utilizing modern engineering tools." [2]

Undergraduate engineering degree programs have adapted to this new reality by including programming as an essential element of first year engineering curricula. This initial experience, however, is not often accompanied by the repeated reinforcement that students in computing disciplines encounter. After their initial exposure, engineering students may not use their programming skills again until much later in their degree program. For example, after their first year introductory course, engineering students exploring mechatronics at our institution may not see coding again until their third year of schooling. Students may easily forget what they have learned and question the value of it, as they do not see it in subsequent courses. This lack of opportunity to utilize newly developed programming skills can strengthen the idea among peer groups that programming is irrelevant to engineers, influencing individual perceptions.

Others have noted problems with student engagement in introductory engineering programming experiences. One study, which examined motivation profiles (including interest/enjoyment, value/usefulness, and perceived choice) of 434 first year engineering students learning programming in a flipped class environment, found that over 80% displayed mid- to low-level motivation for learning programming [3]. This lack of motivation may stem from an engineering identity that does not value programming.

As engineering identity is often formed prior to the choice to major in engineering [4], [5], [6], expansion of the engineering identity to include programming may be needed. While the concept of identity makes several appearances in engineering literature [4], [5], [6], [7], [8], [9], [10], [11], [12], computing literature on identity presents less research [13], [14], [15]. The Engineering Students Attitudes Toward CS Survey [16] is the closest fit in topic but examines attitudes rather than identity.

Our preliminary focus group work suggests that students

tend not to recognize programming as an essential skill within their particular engineering discipline, nor to identify themselves as programmers. Without understanding the value of programming to their identity or careers, they are not motivated to develop their programming skills. These students will remain at a novice level, which may perpetuate the idea that "engineers don't code".

Additionally, our recent survey-based work investigating engineering student attitudes toward computer science suggests that students' self-reported level of understanding of "the way that programs are constructed" is significantly related to both their confidence and intention to take future programming coursework [17]. The factor of interest is a language-independent concept, referring to the understanding of the relationship between textual instructions and corresponding behavior. We use the term *algorithmic thinking* to capture the higher level nature of this factor, lying outside of particular syntactic, semantic, or pragmatic aspects of a given programming context.

Based on this initial work, we hypothesize that two changes could enhance the sense of computing identity among engineering students:

- Better understanding of the utility of programming in the lives of engineers
- Engagement in language-independent problem solving practices, in conjunction with traditional programming exercises

II. Instructional Context

Our team is working to develop and assess an approach that incorporates the algorithmic thinking concepts from our introductory Computer Science (CS) programming course into the first few weeks of the Engineering Fundamentals (EF) program. We have investigated key differences in curricula, located concepts that may benefit from intervention, and are working to mitigate repetition for computer engineering students, who take both courses. The goal is to pilot an approach within the EF program that is effective at promoting understanding and value of programming to engineering students. In this specific work, we focus on an interdisciplinary approach to introductory programming that ensures relevant, meaningful topics are contextually woven into the EF course.

Our approach draws upon current practices employed in our CS program's introductory programming course (CS1), which begins by introducing students to algorithmic thinking. In the first few weeks of class, CS1 covers topics such as: breaking down word problems, identifying key components and programming patterns, communicating with flowcharts and pseudocode, understanding how the computer interprets code, product lifecycle documentation, and debugging. These topics are aimed at students who have never coded before, allowing them to ease into programming by introducing relevant higher level ideas. We couple this with block-based programming and critical thinking activities before moving to a traditional text-based language for the rest of the course.

The first year engineering program (ENG1), in contrast, briefly addresses algorithmic thinking, after which students are immediately introduced to MATLAB by week two. In ENG1, MATLAB is framed as a tool that students can apply to relevant problems. Students learn MATLAB in a flipped learning environment, with support from near-peer teaching assistants. Typical sections of ENG1 have 120 students that meet twice weekly in two-hour "studio sessions" with the instructor and all teaching assistants (TAs), as well as once weekly one-hour sessions with just their TA.

III. Objectives and Interventions

Our primary objective is that ENG1 students will gain a stronger understanding of the value of programming skills in an engineering career. We hope that this fosters a greater appreciation, interest, and intrinsic motivation in learning this material. We also want them to develop greater confidence in their capabilities at approaching problems that require algorithmic conceptualization, with the hope that such confidence fosters continued growth and persistence in programming.

Interventions are targeted for the 2020/21 academic year to achieve the following goals:

1) Establish student awareness that *MATLAB skills can be generalized* to other programming languages, and that conceptual components of program structure are universal across all languages (such as loops, conditional logic, etc.).
2) Incorporate targeted activities for *algorithmic thinking and conceptual design*, directly into existing programming exercises.
3) Demonstrate the *practical application of programming skills* to engineering disciplines to increase engineering student acceptance of programming as useful knowledge.

A. Establishing that MATLAB skills can be generalized

Many core concepts in programming exist across all programming languages in some form. MATLAB is no different: it contains loops, if statements, conditions, variable assignments, structures for holding multiple pieces of data, and more. Our goal with this intervention is to better establish that the core ideas learned by new MATLAB users are able to be generalized to other languages. Within focus groups, our students have presented the misconception that "MATLAB isn't a programming language", and as a result, may not realize the valuable core ideas they gain while learning these concepts.

To target this misconception, we are planning an intervention which will present our students with code in a similar but distinct language from MATLAB that contains core concepts they have already learned in MATLAB. For example, engineering students familiar with MATLAB might be presented with an if statement and logical conditions written in Python. Our goal with this intervention is to pose a working-world scenario where a previous employee has written some code that is required for the team to use, but that the team must decipher what it means.

In reflecting on an industry application where their MATLAB skills may be generalized, we hope that students will better understand the way the skills they have learned apply beyond "just MATLAB". Further, they will see how despite not learning the language in question, they can still identify core components and general ideas of behavior, even if they do not have "mastery" of that language's syntax. This we hope will help students recognize the general applicability of these skills, and that they are not just "MATLAB" skills. Assignments embedded in introductory engineering courses for this purpose will be qualitatively assessed to determine the extent to which students are able to identify core concepts within unfamiliar programming languages.

B. Incorporating targeted activities for algorithmic thinking

A focus of this intervention is to integrate CS approaches to teaching beginning programming into the introductory engineering curriculum. EF has developed preliminary educational scaffolding around algorithmic thinking, but would like to further improve these efforts with the guidance of CS professionals. Currently in ENG1, students have an "Algorithm Template" assignment, where they must plan an algorithm by completing sections of the template. In CS1, students do similar exercises, but with additional guided questions and considerations, aiding in confident development of a strong algorithm template. Additionally, before CS1 students begin working in Java, where they must have an understanding of syntax, they spend about four weeks planning how to approach and structure code. This planning period is spent alongside time working in the visual block-based programming language *Snap!* In contrast, ENG1 students begin class in the first week focused on teaming, communication, ethics, and exploring engineering majors. By the second week of class, they are actively programming in MATLAB, starting by using it as a "calculator" and entering in basic math commands.

We plan to expand the existing algorithm exercise in ENG1 and incorporate an intervention where students are guided through the algorithm template exercise with additional questions and considerations, such as those they might see in CS1. We will review the responses to the guided questions and templates, and also present a quiz as a quantitative assessment to compare intervention and non-intervention groups for overall understanding of algorithmic design. This will likely be conducted among recitation sessions by near-peer teaching assistants, as they would be able to facilitate the activity.

C. Demonstrating the practical application of programming

In the EF context, we need to design our messaging around the value to engineers, use engineering terminology, and incorporate additional concepts in a way that ties to their engineering goals. Messages that are commonly used to encourage the pursuit of computer science may likely not succeed with a student who has developed an engineering identity and does not see any association of that identity with programming. Our messaging and intervention will be designed around showcasing relevance and future value for an engineer.

We are designing interventions specifically targeting the issue of relevance to engineers, which are likely to include presentations or video messages from engineers in industry about how they use programming within their careers. Last year, we implemented this strategy alone, without the additional interventions detailed in this paper. By themselves, presentations on the use of programming skills by practicing engineers did not create measurable impacts on engineering students' acceptance of programming as a useful technology or skill.

To increase engagement with the presentations, we intend to ask students to reflect on the videos, using prompts that will ask them to consider relevancy of skills they are learning to their future engineering careers. Reflective practices have been identified as a valuable pedagogical tool in helping engineering students extract meaning from their experiences [18]. Additionally, ENG1 students will be given an assignment to research and report on examples of how engineers within their intended engineering discipline of study utilize programming. These reflections may be beneficial for qualitative analysis, and the act of writing them may have an affect on student attitudes and perception.

Technology acceptance surveys have already been used in ENG1 for previous data collection, based on the Unified Theory of Acceptance and Use of Technology (UTAUT), and have utility here to measure changes in the students as a result of the interventions [19]. Attitudes toward programming and computer science surveys would also be of value to measure changes in students. In using these surveys throughout the semester, we can gauge what effect, if any, the interventions are having on ENG1 student perception and attitudes.

IV. Implementation Plan

Our interventions are being designed for implementation in one master class of of ENG1 during the Fall 2020 semester. Normally, this class is composed of 5 sections of 24 students, each with their own near-peer TA, for a total of 120 students. The course is taught with a flipped-class class structure, employing a combination of online pre-lesson activities, once-per-week individual section sessions led by near-peer TAs, and twice-per-week studio sessions with the full class of five sections. It is anticipated that this course will be offered at least partially online in the Fall of 2020 due to COVID-19. The ENG1 master class size will be reduced to 100 students in 5 sections of 20, each under the supervision of a near-peer TA. Three of these sections will be assigned to the intervention condition; two will serve as the control. Semiweekly studio sessions with the full class of five sections will be held through videoconferencing. After a short introduction to the day's materials by the course instructor, breakout rooms will be utilized for the active learning work of student team of 4. The instructional team, including the instructor and all near-peer-TAs, will circulate throughout the breakout rooms, responding to student questions as needed. A group chat will

be used by the instructor and TAs to communicate about which teams need instructional attention. The once-per-week individual section sessions, follow a similar Zoom format, utilizing breakout rooms. However, the teaching staff will be solely comprised of a single near-peer TAs. Intervention activities (as detailed in Section III) will be delivered primarily through the online, pre-lesson activities and in the small group meetings with the near-peer TAs.

A. Timeline

Spring 2020. To date, we have focused on comparing the algorithmic thinking curriculum in CS1 and ENG1 and identifying opportunities for infusing CS1 concepts into the ENG1 course. Additionally, we have investigated pain points in ENG1 regarding student perception, as well as touch points to CS concepts. We have drafted our initial approach and are seeking feedback and additional ideas.

Summer 2020. Prior to fall semester, we will refine our implementation plan and prepare for deployment. Our team will identify specific points within ENG1's schedule for generalization of skills, targeted activities, and demonstration of practical application. We will create training materials supporting TA delivery of intervention activities and data collection.

Fall 2020. Interventions will be employed in ENG1. Interventions (detailed in Section III) will be administered within three sections of the course, with two sections utilized as controls. Interventions will be administered either within the pre-lessons or the TA sessions. Assessments, as well as any additional data collection, will be conducted. Any modifications required to the process over the course of the semester will be documented. The control and intervention sections of the ENG1 course will allow for comparison.

Spring 2021. We will conduct a full analysis of data and assessments from Fall 2020. Findings and process will be reported. We will consider whether broader implementation and validation of the intervention is warranted for the next academic year, or if new approaches should be adopted.

B. Assessment

Technology Acceptance and Attitude Surveys. Our previous work has explored student responses throughout the semester to these surveys. Surveys examining student attitudes and acceptance [19], [16] can allow us to recognize if students perceive that MATLAB skills can generalize to other technology, and if they are more accepting of MATLAB as a programming language. Also, as we have given such surveys previously, we will not only be able to compare this semester, but potentially compare to previous semesters as well.

Responses to Algorithm Worksheets and Quiz. Student responses to ENG1 algorithm exercises can be qualitatively investigated for correctness and clarity of responses: specifically, do students who received the intervention complete the worksheets with greater precision than the control? In addition, the quiz we develop on this topic will allow for more targeted exploration of the data, and quantitative measures to inform our qualitative exploration.

Review of Reflections. Student responses to the reflection prompts on engineering careers and relevant skills may also provide valuable qualitative feedback to further our understanding of engineering student attitudes, as well as to assess what impacts the intervention may have had.

V. Impact

Success in this project would see students in ENG1 having an increase in confidence, interest, and understanding of programmatic thinking in an engineering context, and the value it can have for their future careers. We hope to learn what works well, find new ideas to try moving forward, and identify what may not work as anticipated regarding integrating disciplinary approaches across courses.

This work is aligned with our university's 21st Century Education Initiative, which entails preparing our students for the Fourth Industrial Revolution. This requires all students understand the digital landscape that is now an everyday component of our world. We hope to find ways to more strongly connect our ENG1 course to these ideas, and would enjoy seeing such efforts ripple across courses and majors. We hope the path we chart here can serve as a model for further interdisciplinary work within our university.

In addition, we hope our plan, findings, and reflection can guide courses outside of our university in designing multidisciplinary infusions. Use of computation is a reality in careers across majors around the world. Increasing student acceptance and knowledge is an imperative for their success beyond just our university. Our work aims to encourage adoption of relevant interventions beyond our university. In addition, we hope that with successful interventions, replication and cross-university studies could be conducted as well.

VI. Conclusion

Our work in progress is exploring the infusion of introductory computer science principles into an engineering fundamentals curriculum. We hope to motivate students to recognize that MATLAB programming skills can be generalized, to better target activities surrounding algorithmic thinking, and to demonstrate the practical applications of programming skills in engineering careers. Our ENG1 students possess engineer identities, but do not often see programming skills as relevant to this. Our interventions are intended to modify student acceptance, utility, and understanding of the value of programming in engineering contexts. We hope this approach can later be scaled and applied to other disciplines working to add computing concepts into their curriculum.

References

[1] J. Kerns, "Do mechanical engineers need programming to survive?" *Machine Design*, March 2017. [Online]. Available: https://www.machinedesign.com/community/editorial-comment/article/21835239/do-mechanical-engineers-need-programming-to-survive

[2] "ABET engineering accreditation criteria," 2020. [Online]. Available: https://www.abet.org/accreditation/accreditation-criteria/criteria-for-accrediting-engineering-programs-2020-2021/

[3] L. M. Lingar, A. Williams, I. Jeldes, and R. McCord, "Motivation profiles of non-major computer programmers in a flipped classroom environment," in *American Society for Engineering Education (ASEE) First Year Engineering Experience (FYEE) Conference*. Daytona Beach, Florida: ASEE Conferences, August 2017. [Online]. Available: https://peer.asee.org/29427

[4] A. Godwin, G. Potvin, Z. Hazari, and R. Lock, "Identity, critical agency, and engineering: An affective model for predicting engineering as a career choice," *Journal of Engineering Education*, vol. 105, no. 2, pp. 312–340, 2016. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/jee.20118

[5] P. Vincent-Ruz and C. Schunn, "The nature of science identity and its role as the driver of student choices," *International Journal of STEM Education*, vol. 5, 12 2018.

[6] B. M. Capobianco, B. F. French, and H. A. Diefes-Du, "Engineering identity development among pre-adolescent learners," *Journal of Engineering Education*, vol. 101, no. 4, pp. 698–716, 2012. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/j.2168-9830.2012.tb01125.x

[7] J. R. Morelock, "A systematic literature review of engineering identity: definitions, factors, and interventions affecting development, and means of measurement," *European Journal of Engineering Education*, vol. 42, no. 6, pp. 1240–1262, 2017. [Online]. Available: https://doi.org/10.1080/03043797.2017.1287664

[8] A. Godwin, "The development of a measure of engineering identity," *ASEE Annual Conference & Exposition*, Jan 2016. [Online]. Available: http://par.nsf.gov/biblio/10042227

[9] D. T. Rover, "Engineering identity," *Journal of Engineering Education*, vol. 97, no. 3, pp. 389–392, 2008. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/j.2168-9830.2008.tb00986.x

[10] D. M. Hatmaker, "Engineering identity: Gender and professional identity negotiation among women engineers," *Gender, Work & Organization*, vol. 20, no. 4, pp. 382–396, 2013. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1468-0432.2012.00589.x

[11] K. Tonso, *Engineering identity*, January 2015, pp. 267–282.

[12] K. Meyers, *Engineering identity as a developmental process*. Purdue University, 2009.

[13] J. Mahadeo, Z. Hazari, and G. Potvin, "Developing a computing identity framework: Understanding computer science and information technology career choice," *ACM Trans. Comput. Educ.*, vol. 20, no. 1, Jan. 2020. [Online]. Available: https://doi.org/10.1145/3365571

[14] A. N. Washington, S. Grays, and S. Dasmohapatra, "The computer science attitude and identity survey (csais): A novel tool for measuring the impact of ethnic identity in underrepresented computer science students," 2016.

[15] A. Garcia, M. Ross, Z. Hazari, M. Weiss, K. Christensen, and M. Georgiopoulos, "Examining the computing identity of high-achieving underserved computing students on the basis of gender, field, and year in school," *Collaborative Network for Engineering and Computing Diversity (CoNECD)*, Apr 2018. [Online]. Available: http://par.nsf.gov/biblio/10075621

[16] A. Hoegh and B. M. Moskal, "Examining science and engineering students' attitudes toward computer science," in *2009 39th IEEE Frontiers in Education Conference*, 2009, pp. 1–6.

[17] K. Steelman, M. Jarvie-Eggart, K. Tislar, N. Manser, B. Bettin, L. C. Ureel, and C. Wallace, "(work in progress: The perception of computer programming within engineering education: An investigation of student attitudes, beliefs, and behaviors," in *To Appear In: 2020 ASEE Annual Conference & Exposition*, 2020, pp. xxx–xxx.

[18] J. Turns, B. Sattler, K. Yasuhara, J. Borgford-Parnell, and A. J., "Integrating reflection into engineering education," vol. 35, 06 2014, p. 64.

[19] V. Venkatesh, M. G. Morris, G. B. Davis, and F. D. Davis, "User acceptance of information technology: Toward a unified view," *MIS Quarterly*, vol. 27, no. 3, pp. 425–478, 2003. [Online]. Available: http://www.jstor.org/stable/30036540