Computational Thinking Growth During a First-Year Engineering Course

Noemi V. Mendoza Diaz
Dept. of Educational Administration and Human Resource Development
Texas A&M University
College Station, TX, USA
nmendoza@tamu.edu

Russ Meier
Dept. of Electrical Engineering and Computer Science
Milwaukee School of Engineering
Milwaukee, WI, USA
meier@msoe.edu

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

So Yoon Yoon
Dept. of Engineering Education
University of Cincinnati
Cincinnati, OH, USA
yoons5@ucmail.uc.edu

***Abstract*— This full research-track paper demonstrates growth in computational thinking in a cohort of engineering students completing their first course in engineering at a large Southwestern university in the United States. Computational thinking has been acknowledged as a key aspect of engineering education and an intrinsic part of multiple ABET outcomes. However, computing is an area where some students have more privileges (e.g. access and exposure to meaningful use of computers) than others. Integrating computing into engineering, especially early in the curriculum, may exacerbate existing experiential disadvantages students from excluded social identities experience. Most introductory engineering programs have a component of programming and/or computational thinking. A comprehensive literature review showed that no existing computational thinking framework fully met the needs of students and professors in engineering and computer science. As a result, this team created the Engineering Computational Thinking Diagnostic (ECTD). This diagnostic was assessed and improved during the 2019-2020 academic year. Data was collected from a cohort in a first-year engineering course that included topics in mathematics, engineering problem solving, and computation. Pre- and post-test data analysis with 62 participants documents statistically significant student growth in computational thinking in this course. Significant differences were not found by gender or a limited racially-based analysis. This diagnostic is of interest and relevance to all institutions providing engineering and computing programs. The short-term impact of this research includes an innovative approach to gauge student abilities in computational thinking early in a course in order to add appropriate intervention activities into lesson plans. The long-term impact is the creation of a measurement of student learning of computational thinking in engineering for courses and programs that wish to develop this important skill in their students.**

Keywords—computational thinking, first-year engineering education, learning and teaching effectiveness, assessment metrics.

I. Introduction And Research Motivation

Undergraduate engineering and computer science students spend at least four years learning the theories, techniques, and skill sets required to practice their chosen field. The acquisition of this technical knowledge is accompanied by a gradual self-actualization of belonging in the community of professionals. This enculturation into the engineering and computer science professions differs for every student because each student has unique life experiences. Demographics, socioeconomic factors, and entry-level preparedness all contribute to the speed and success of enculturation.

One of the critical points in engineering and computer science enculturation is the first-year experience. Research at a large institution in the United States discovered many factors affecting student success and enculturation during the first year of engineering education [1]. The key results of that study were that struggling students found computational thinking particularly challenging, tasks related to computational thinking demotivated them, and tasks related to computational thinking made them question their choice of major. Many engineering and computer science curricula include foundations of computation or computer programming during the first year. Yet, the connection between enculturation and these computing courses is not well-documented in the literature.

The evident gap in this area of research, along with the results from the initial study, led to a broader multi-institutional research project into how computational thinking affects the formation of engineers. With funding from the United States National Science Foundation, a team of researchers is working to understand the multiple factors that affect computational thinking development and improve the way computational thinking is taught to engineers at the college level. The research questions addressed in this paper are:

1. *What are the learning gains of students who attended an introductory first-year engineering course that includes computation thinking topics when assessed by the ECTD?*
2. *How do the learning gains of students differ by gender and race?*

To help answer these questions, the research project has developed and deployed an instrument called the Engineering Computational Thinking Diagnostic (ECTD) that gauges growth in computational thinking. This diagnostic can be used by an instructor as a pre-test to determine both skills and deficits in a group of students beginning a first course in computation. At the end of the course, the instructor can use the instrument as a post-test comparison tool to measure growth in computational thinking. This paper demonstrates the use of the diagnostic at a large university in the United States to gauge the development of computational thinking skills in one cohort of first-year engineering students.

This material is based upon work supported by the National Science Foundation under Grant No. 1949880

II. Literature Review

National and international reports and publications corroborate the importance of computation in the formation of engineers [2-9]. Of course, the development of computational thinking skills is also of primary importance in computer science students.

Computation skills are also acknowledged universally, beyond the engineering and technical fields, to the extent that several pre-university initiatives have been launched. Example K-12 programs including fundamentals of computation or programming include Hour of Code, Code.org, Project Lead the Way, Girls Who Code, and Black Girls Code. And, a recent United States presidential initiative, Computer Science for All, was announced and funded [10-11]. Capacity to code is directly linked to technological independence and advancement. It is generally believed that it should start as early as possible in the educational pathway.

The literature also reveals a disparity in representation of a variety of social identity groups in the fields of computer science and engineering. Margolis and Fisher exposed factors that caused women to be underrepresented at Carnegie Mellon University [12]. Underrepresentation is also prevalent in racial and ethnic minority groups and those from low socioeconomic status, [13-15]. The challenge of building computational skills in all areas of human initiative and specifically in engineering is not simple.

Computing, coding, programming, algorithmic thinking and computational thinking are all terms used in the engineering and computer science education literature. Authors link these concepts with technological and engineering knowledge advancement [3, 9, 16-21]. The relationship between engineering education and computing skills is formalized in The Taxonomy of Engineering Education Research where "Computing skills (syn: Computing knowledge)" are included in the student outcome category [22]. The most prevalent reported relationships between engineering and all these terms are in the context of problem solving, systems thinking, modeling, simulation, and design [2, 4, 23-28]. However, the way engineers understand computational thinking evinced by these papers differs from the frameworks in computational thinking in the literature.

Defining computational thinking appropriate to an engineering context is surprisingly difficult. Most computational thinking frameworks suggested by different groups and researchers do not specifically target engineering. For example, the International Society for Technology in Education (ISTE) and the Computer Science Teachers Association (CSTA) both have well designed and vetted frameworks, but these frameworks were designed for K-12 teaching [29, 30]. Brennan and Resnick also defined a computational thinking framework during their work with Scratch (a graphical programming language widely used in K-12) for the MIT Media Lab [31]. However, the concepts in this framework are centered on programming. While programming is a powerful tool for engineering, computational thinking is broader than programming [32].

The Collaborative Process to Align Computing Education with Engineering Workforce Needs (CPACE) defined computing needs based on current engineering workforce needs. The project stakeholders were from one geographic region of the United States and included community colleges, industry, researchers, engineering faculty, engineering students and business leaders [4]. While their work is at the college level and centered on engineering, it combined information technology and computational thinking and is not focused on a broad view of developing computational thinking skills as part of enculturation into the engineering profession. A framework for computational thinking is also not a diagnostic test for computational thinking. These frameworks have learning objectives, but lack a validated testing mechanism.

The most prominent existing test for computational thinking is the Computer Science Principles (CSP) examination. The College Board, a non-profit organization headquartered in the United States, offers the CSP. The CSP has a well-designed computational thinking framework as well as a diagnostic examination; however, it was designed to represent general education computer science at the college level and is not specifically responsive to the needs of engineering [33]. General college students and engineering college freshman have different backgrounds, especially in mathematics. The prerequisite mathematics for the CS Principles examination is given below [34]. Students with only this mathematical background would be unlikely to pursue, let alone succeed, in engineering. Any diagnostic test for a general audience will have to avoid using the mathematically intense parts of computation that engineers most need.

> *"…a student in the AP Computer Science Principles course should have successfully completed a first-year high school algebra course with a strong foundation in basic linear functions and composition of functions…In addition, students should be able to use a Cartesian (x,y) coordinate system to represent points in a plane." [34, pp. 4]*

The CSP is part of a system of examinations that are used by secondary students to gain college credit in the U.S. The examination is lengthy and expensive. The CSP consists of a portfolio of two performance tasks that are developed over the course of a year and a two-hour multiple-choice examination with seventy-four questions [35]. The examination costs between $94 and $182, depending on where the examination is taken and when it is scheduled [36]. Students that are enrolled in college are not permitted to take the CSP examination. Therefore, the CSP cannot be used to gauge computational thinking among engineering students.

In order to develop and deploy a survey that educators could use to gauge computational thinking in engineering students, a framework for computational thinking was developed that built upon aspects of earlier work and emphasized aspects recognized by the expertise of professors in the area of engineering education and computer science who had taught computational thinking and programming to engineers for several years.

The framework in Table I underlies the ECTD, which incorporates five aspects of computational thinking: (a) Abstraction, (b) Algorithmic Thinking and Programming, (c) Data Representation, Organization, and Analysis, (d) Decomposition, and (e) Impact of Computing. Table I delineates how the most recent ABET student outcomes from 2019-2020 are aligned with the framework in the literature as well as the framework for the ECTD. Key terms are defined as below.

- ***Abstraction***: A new representation of a thing, a system, or a problem that reframes a problem by hiding details irrelevant to the question at hand.
- ***Algorithmic Thinking and Programming***: Developing systematic methods to solve problems and expressing these methods in an appropriate language.
- ***Data Representation, Organization, and Analysis***: Transforming raw data into information and knowledge.
- ***Pattern Matching***: Finding similarities between data or algorithms.
- ***Automation***: Plugging pieces into an algorithm to help with a result, sometimes involving programming.
- ***Decomposition***: Breaking a problem or system apart into smaller components that can be more easily and completely analyzed.
- ***Impact of Computing***: Considering both the potential harm and benefits to multiple groups when making computing choices and decisions.

TABLE I. ENGINEERING COMPUTATIONAL THINKING DIAGNOSTIC (ECTD) FRAMEWORK COMPARED WITH THE LITERATURE

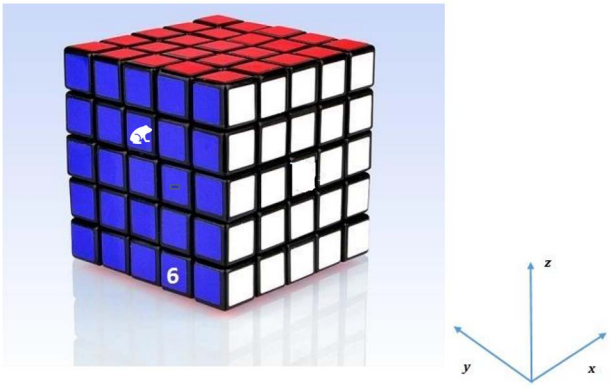
<table>
<tr><th>ECTD</th><th>ABET [37]*</th><th>ISTE [29]</th><th>CSP [33]</th><th>CPACE [4]</th><th>CSTA [30]</th><th>Brennan & Resnick [32]</th></tr>
<tr><td>Abstraction</td><td>1, 7</td><td>Abstraction</td><td>Abstraction</td><td>Modeling and Abstraction</td><td></td><td></td></tr>
<tr><td rowspan="4">Algorithmic Thinking and Programming</td><td rowspan="2">6, 7</td><td></td><td>Algorithms</td><td rowspan="2">Algorithmic Thinking and Programming</td><td rowspan="2">Algorithms and Programming</td><td></td></tr>
<tr><td></td><td>Programming</td><td>Multiple Programming Constructs e.g. loops, selection</td></tr>
<tr><td>1, 7</td><td>Pattern Matching</td><td></td><td></td><td></td><td></td></tr>
<tr><td>1, 7</td><td>Automation</td><td></td><td></td><td></td><td></td></tr>
<tr><td rowspan="2">Data Representation, Organization, and Analysis</td><td rowspan="2">6, 7</td><td></td><td rowspan="2">Data and Information</td><td>Digital Representation of Information</td><td rowspan="2">Data and Analysis</td><td rowspan="2">Data</td></tr>
<tr><td></td><td>Information Organization</td></tr>
<tr><td>Decomposition</td><td>1, 7</td><td>Decomposition</td><td></td><td></td><td></td><td></td></tr>
<tr><td rowspan="2">Impact of Computing</td><td>2</td><td></td><td>Global Impact</td><td></td><td>Impacts of Computing</td><td></td></tr>
<tr><td>2</td><td></td><td>The Internet</td><td>Networks</td><td>Networks & the Internet</td><td></td></tr>
</table>

*The ABET outcomes are: 1. an ability to identify, formulate, and solve complex engineering problems by applying principles of engineering, science, and mathematics; 2. an ability to apply engineering design to produce solutions that meet specified needs with consideration of public health, safety, and welfare, as well as global, cultural, social, environmental, and economic factors; 6. an ability to develop and conduct appropriate experimentation, analyze and interpret data, and use engineering judgment to draw conclusions; 7. an ability to acquire and apply new knowledge as needed, using appropriate learning strategies.

ISTE = International Society for Technology in Education; CSP = Computer Science Principles; CPACE = Collaborative Process to Align Computing Education with Engineering Workforce Needs; CSTA = Computer Science Teachers Association

The Engineering Computational Thinking Diagnostic operationalizes all these aspects of computational thinking in two versions of the diagnostic of 15 question-items, each with five multiple choice options. Two example items from the ECTD are given below, with correct answers in bold. The first focuses on decomposition and the second focuses on the impact of computing.

Decomposition: The Rubik's cube in the figure is composed of 5^3 blocks. A program counts the number of blocks traversed from the origin to the desired block by first traversing along the x-axis, then the y-axis, and finally the z-axis. For example, the block labeled 6 is the 6^{th} block accessed. How many blocks are traversed to get to the block with the frog icon?



a) 6 b) 11 c) 18 d) 78 **e) 86**

Fig. 1. An example of decomposition item question in the ECTD

Students can answer this question by decomposing the problem into three parts: counting the number of 25 block planes parallel to the *x-y* axis of the cube that have been traversed, then adding in the number of 5 block rows parallel to the *x* axis that have been traversed, and lastly adding in the last individual block. The distractors were designed to identify students who traversed the block in orders other than those requested. For example, if you traverse the first three levels of 25 blocks, and then traverse along the *y* axis instead of the *x* as first, you will touch 78 blocks (answer *d*). We made the correct answer one of the extremes to keep the student trick of avoiding choosing extreme answers from being successful.

Impact of Computing: You have designed a system called TestFile that allows students at your university to share examinations from previous semesters of the same class with the same professor. Users pay $50 for this service or have to upload at least ten examinations from their previous semester classes. Which of the following statements is true?

a) This system is fair to students who are in their first semester at your institution because they can always pay $50 if they don't have examinations to share.
b) This system respects the intellectual property of the professors who teach at your institution, since the graded examinations that are given back to students are the student's personal property.
c) This system is unfair to students who are in their first semester at your institution because they do not have examinations to share and would have to pay for the service or have an academic disadvantage.
d) You could not be charged with academic misconduct for setting this system up at your own institution because it's a free country.
e) Students who use your system could not be charged with academic misconduct because you have made the system available to everyone in the class.

This problem addresses a number of issues in the social impact of computing, including unequal access to resources (money, time in higher education), intellectual property ownership, and academic integrity. Each distractor is based on a misunderstanding of one of these critical elements. For example, choice a) would be chosen by a student who didn't realize that not everyone has an extra $50 to pay for educational resources. Answer b) would be chosen by a student who didn't understand that people who create intellectual property (like a professor's examinations) may have the right to control distribution beyond the intended audience.

The purpose of this study is to present the preliminary results of the application of this diagnostic to first-year engineering students before and after an introductory engineering course. As stated earlier, the research questions guiding this research paper are:

1. *What are the learning gains of students who attended an introductory first-year engineering course that includes computation thinking topics when assessed by the ECTD?*
2. *How do the learning gains of students differ by gender and race?*

III. Method

Participants. The population of the study was the entire 2019 first-year engineering cohort at a large Southwestern university in the United States. Institutional Review Board (IRB) approval was obtained and participants were recruited via IRB-approved emails. To avoid participation bias, recruitment emails were distributed to all first-year engineering students by faculty members not affiliated with the research team. Pre-test recruitment emails were sent twice in the second week of the Fall semester. About 800 students responded to the pre-test version of the ECTD. After factor analysis to gauge the effectiveness of the ECTD questions, changes to questions were made before post-test use of the ECTD. At the start of the Spring semester, two recruitment emails were sent to recruit participants from the same population for post-test measurement. While around 100 students responded to the post-test version of the ECTD, 62 students were identified for completion of both the pre-post ECTD. Table II shows demographic information of these 62 participants.

TABLE II. Participants' Demographics

Gender	Race and Ethnicity						Sub-total
	Hispanic	*AIAN*	*Asian*	*Black*	*Multiracial*	*White*	
Female	1	1	3	0	3	10	18
Male	7	0	10	1	0	26	44
Total	**8**	**1**	**13**	**1**	**3**	**26**	**62**

Note. AIAN = American Indian or Alaska Native

Data Analysis. Based on the sample size, the research team conducted a paired sample *t*-test or Wilcoxon signed ranks test, which is a counterpart of the paired sample *t*-test in nonparametric tests, to determine if there is a significant difference between the pre- and post-ECTD results. Due to the small sample size by subgroups of gender and minority status, we conducted a Mann-Whitney U test, which is a counterpart of independent samples *t*-test in nonparametric tests, to explore subgroup differences in the pre-post changes on the ECTD scores. Here, minority status was grouped as White versus non-White students due to the small sample size.

IV. Results

Table III shows the descriptive statistics on the pre-post ECTD scores by gender and minority status. The differences between the pre-post mean scores were all positive, indicating the gain of student learning on computational thinking.

TABLE III. DESCRIPTIVE STATISTICS ON THE PRE-POST ECTD SCORES

Subgroup	N	Pre-score		Post-score		Mean Gain
		M	SD	M	SD	
Female	18	7.89	2.54	11.67	2.03	3.78
Male	44	8.95	2.93	11.16	2.97	2.21
Non-White	26	8.54	3.37	10.77	3.29	2.23
White	36	8.72	2.43	11.69	2.20	2.97
Total	**62**	**8.65**	**2.84**	**11.31**	**2.73**	**2.66**

Table IV shows the statistical testing results on the pre-post differences by gender and minority status. The paired sample *t*-test indicates a significant mean difference of 2.26 between pre and post-ECTD scores with $t(61) - 7.4$, $p < 0.001$ with a significant correlation of $r = 0.480$ between pre-post scores. Similarly, paired sample *t*-test and Wilcoxon signed ranks tests shows significant increases on the post-scores for each subgroup by gender and minority status. The Cohen's *d* effect sizes on the pre-post changes were all positive and medium to large [38]. Perceptually, the magnitudes of the change were larger for female and White students.

TABLE IV. STATISTICAL TESTING ON PRE-POST ECTD SCORES

Subgroup	N	Paired Sample t-test			Wilcoxon Signed Ranks Test		Cohen's d
		t	df	p	Z	p	
Female	18	–	–	–	-3.635	< 0.001	1.637
Male	44	-5.3	43	< 0.001	–	–	0.747
Non-White	26	–	–	–	-3.003	0.003	0.669
White	36	-7.6	35	< 0.001	–	–	1.278
Total	**62**	**-7.4**	**61**	**< 0.001**	–	–	**0.956**

On average, female students increased 3.78 points and male students increased 2.21 points on the post-ECTD. However, the Mann-Whitney *U* test shows nonsignificant difference by gender on the changes of the pre-post scores, with $U = 301.0$, $Z = -1.489$, $p = 0.137$. Similarly, non-White students increased 2.23 and White students increased 2.97 points on the post-ECTD. However, the Mann-Whitney *U* test shows nonsignificant difference by gender on the changes of the pre-post scores, with $U = 375.5$, $Z = -1.332$, $p = 0.183$.

V. Discussion, Conclusions, and Future work

The research team applied the Engineering Computational Thinking diagnostic in order to partially answer the research question *What are the learning gains of students who attended an introductory first-year engineering course that includes computation thinking topics when assessed by the ECTD?* Students spend four years in most university engineering curricula and growth in computational thinking may occur across all four years. The application of the ECTD at the start of the first year of engineering yields an assessment outside of traditional course assignments to help identify students at-risk of failure in first-year engineering computation courses and to measure growth in the cohort of students that completed the course by comparing pre- and post-test performance.

This paper demonstrates that the first-year engineering course did result in engineering students increasing their computational thinking skills in a statistically significant way. While this result is not surprising, it does provide some verification both of student learning and of the ECTD's ability to detect student learning. If the statistical results had not been significant, either the course's ability to deliver student learning in computational thinking or the ECTD's ability to identify student learning in computational thinking would have been suspect.

The absence of statistically significant differences between male and female students and White and non-White students could have several interpretations. A possible interpretation is that the ECTD is not biased for or against male, female, White or non-White students. It was somewhat surprising, however, that statistically significant differences in computational thinking gains were not seen along either gendered or racial lines, especially given the well-known inequities in computational experience that have historically favored males over females and Whites over non-Whites [12,14].

While the differences in group means were not significant, it is notable that female students seemed to gain more in computational thinking than male students, both starting with lower scores and ending with higher scores. Starting with lower scores would be predicted by the literature [12]. The pattern in White versus non-White students was different. Non-white students started with lower scores than white students, as would be predicted given societal inequities and the literature, but also ended with lower scores [14]. Since non-White students more often come from lower socioeconomic groups, it is possible that non-White students have more competing responsibilities (e.g. family obligations, work) that keep them from making the same gains in computational thinking [39]. There also could be other unidentified social processes at work in the engineering course. It is possible that these results might be statistically significant had the sample size been larger. As such, readers should treat these results as

preliminary until the larger studies that are planned in the near future can be completed and published.

Unfortunately, Spring 2020 proved to be a challenging semester not only at the institution where data was collected but worldwide due to the Covid-19 global pandemic. This impaired the ability of the team to gain sufficient participants to make broader claims about which engineering students benefit the most from computational thinking instruction. For example, comparisons by other social identities, including intersectional identities, were planned, however the sample size was insufficient. The results that are presented should be viewed as preliminary until a larger sample data from the fully validated instrument can be collected and analyzed.

The availability of the ECTD provides an avenue for engineering programs to evaluate how the integration of computational thinking is impacting their programs. As an example, consider how the integration of computational thinking might impact program diversity and inclusion efforts. While the claim that computational skills are necessary for future engineers is not controversial, there are risks to including computational thinking early in engineering programs. Computing is an area of great inequality, where students with greater privileges (who are more likely to be from dominant social groups) have substantially more opportunities to gain pre-university experience with computers and computational thinking [12, 14]. Prior experience with computing may give students from the dominant social groups further advantages. Computing is, after all, one of the least diverse disciplines in all of academia for many reasons. It is therefore possible that integrating computational thinking into introductory engineering classes, may exacerbate the substantial challenges that engineering programs have in attracting and retaining more diverse students.

In order to address this issue, much larger scale experiments will need to be run, especially at large institutions and institutions with greater diversity. These experiments will enable the evaluation of the ways in which computational thinking skill development is influenced by a student's gender, race/ethnicity, disability status, socioeconomic status, and other social identities.

In time, this team hopes that the availability of the ECTD will make it possible for many institutions to systematically examine alternatives for integrating computational thinking into engineering and other related disciplines. Questions that could be considered include:

- In what ways do students who are taught computational thinking in lower division classes learn compared to students who are taught in upper division classes? For example, do students who have had calculus learn computational thinking more effectively than those with lower levels of demonstrated mathematical achievement?
- In what ways can the ECTD provide insight about the learning of computational thinking in areas related to engineering (such as science, technology and mathematics) and in different engineering majors? For example, in what ways do environmental engineers learn computational thinking compared to electrical engineers? Can ECTD results be used to inform individual students which pathways in engineering are more likely to be successful?
- In what ways can the results of the ECTD, when used as a pretest, inform instruction in computational thinking? Is it possible to use the ECTD to identify areas where particular groups of students may struggle in advance and make teaching decisions that lead to better student outcomes?
- In what ways could the ECTD provide information about how effective individual instructors are at developing computational thinking skills in students? In this way, the ECTD could even be used as one component in the evaluation of teaching. This analysis could lead to the identification of patterns of teaching that lead to success in computational thinking for students.

REFERENCES

[1] Mendoza Diaz N.V., Yoon Y. S., Richard J., "Exploring Enculturation in the First-Year Engineering Program (Year III)" In Proceedings of the American Society for Engineering Education, *2019*. Tampa, Florida, USA.

[2] Sorby, S. A., & Baartmans, B. J., "The Development and Assessment of a Course for Enhancing the 3-D Spatial Visualization Skills of First Year Engineering Students", Journal of Engineering Education, 89(3), pp. 301-307, 2000.

[3] Thuné, M., & Eckerdal, A., "Variation theory applied to students' conceptions of computer programming", European Journal of Engineering Education, 34(4), pp. 339-347, 2009.

[4] Vergara, C. E., Urban-Lurain, M., Dresen, C., Coxen, T., MacFarlane, T., Frazier, K., ... & Sticklen, J., "Aligning computing education with engineering workforce computational needs: New curricular directions to improve computational thinking in engineering graduates", In Frontiers in Education Conference, 2009.

[5] Magana, A. J., Brophy, S. P., & Bodner, G. M., "Instructors' intended learning outcomes for using computational simulations as learning tools", Journal of Engineering Education, 101(2), pp. 220-243, 2012.

[6] Gross, S., Kim, M., Schlosser, J., Lluch, D., Mohtadi, C., & Schneider, D., "Fostering computational thinking in engineering education: Challenges, examples, and best practices", In Global Engineering Education Conference (EDUCON), pp. 450-459, 2014.

[7] Meyer, M., & Marx, S., "Engineering dropouts: A qualitative examination of why undergraduates leave engineering", Journal of Engineering Education, 103(4), pp. 525-548, 2014.

[8] Magana, A. J., Falk, M. L., Vieira, C., & Reese, M. J., "A case study of undergraduate engineering students' computational literacy and self-beliefs about computing in the context of authentic practices. "Computers in Human Behavior, 61, pp. 427-442, 2016.

[9] Xia, B. S., & Liitiäinen, E., "Student performance in computing education: an empirical analysis of online learning in programming education environments", European Journal of Engineering Education, 42(6), pp. 1025-1037, 2017.

[10] The White House, "Computer Science of All", Retrieved from https://obamawhitehouse.archives.gov/blog/2016/01/30/computer-science-all, Accessed 5-29-2020.

[11] Romm, T., "Recode: President Trump and his daughter Ivanka are unveiling a new federal computer science initiative with major tech backers", Retrieved from https://www.recode.net/2017/9/25/16276904/president-donald-trump-ivanka-tech-stem-computer-science-coding-education-amazon-google, Accessed 5-29-2020.

[12] Margolis, J., & Fisher, A., "Unlocking the clubhouse: Women in computing", MIT Press, 2003.

[13] Buzzetto-More, N. A., Ukoha, O., & Rustagi, N., "Unlocking the barriers to women and minorities in computer science and information

systems studies: Results from a multi-methodolical study conducted at two minority serving institutions", Journal of Information Technology Education: Research, Volume 9, pp. 115-131, 2010.

[14] Margolis, J., "Stuck in the shallow end: Education, race, and computing", MIT Press, 2010.

[15] Varma, R., "Making computer science minority-friendly", Communications of the ACM, 49(2), pp. 129-134, 2006.

[16] Blikstein, P., Kabayadondo, Z., Martin, A., & Fields, D., "An assessment instrument of technological literacies in makerspaces and FabLabs", Journal of Engineering Education, 106(1), pp. 149-175, 2017.

[17] Chan, C.K., Zhao, Y. and Luk, L.Y., "A validated and reliable instrument investigating engineering students' perceptions of competency in generic skills", Journal of Engineering Education, 106(2), pp. 299-325, 2017.

[18] Shyamala, C. K., Velayutham, C. S., & Parameswaran, L., "Teaching computational thinking to entry-level undergraduate engineering students at Amrita University", In Global Engineering Education Conference (EDUCON), pp. 1731-1734, 2017.

[19] Nelson, K. G., Shell, D. F., Husman, J., Fishman, E. J., & Soh, L. K., "Motivational and self‐regulated learning profiles of students taking a foundational engineering course", Journal of Engineering Education, 104(1), pp. 74-100, 2015.

[20] Walker, H. M., "Computational thinking in a non-majors CS course requires a programming component", ACM Inroads, 6(1), pp. 58-61, 2015.

[21] Lowe, D. B., Scott, C. A., & Bagia, R., "A skills development framework for learning computing tools in the context of engineering practice", European Journal of Engineering Education, 25(1), pp. 45-56, 2000.

[22] Finelli, C. J., "EER Taxonomy (Version 1.1)", Retrieved from http://taxonomy.engin.umich.edu/wp-content/uploads/2018/05/EER-Taxonomy-Version-1.1.pdf, Accessed 5-29-2020.

[23] Hacker, M., "Integrating Computational Thinking into Technology and Engineering Education", Technology and Engineering Teacher, 77(4), pp. 8-14, 2018.

[24] Magana, A. J., Falk, M. L., Vieira, C., Reese Jr, M. J., Alabi, O., & Patinet, S., "Affordances and challenges of computational tools for supporting modeling and simulation practices", Computer Applications in Engineering Education, 25(3), pp. 352-375, 2017.

[25] Cooper, S., & Dann, W., "Programming: a key component of computational thinking in CS courses for non-majors", ACM Inroads, 6(1), 50-54, 2015.

[26] Miller, L. D., Soh, L. K., Chiriacescu, V., Ingraham, E., Shell, D. F., & Hazley, M. P., "Integrating computational and creative thinking to improve learning and performance in CS1", In Proceedings of the 45th ACM Technical Symposium on Computer Science Education, pp. 475-480, 2014.

[27] Wing, J. M., "Computational thinking and thinking about computing", Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences, 366(1881), pp. 3717-3725, 2008.

[28] Yasar, O., & Landau, R. H., "Elements of computational science and engineering education", SIAM review, 45(4), pp. 787-805, 2003.

[29] Krauss, J., & Prottsman, K., "Computational Thinking and Coding for Every Student: The Teacher's Getting-Started Guide", Corwin Press, 2016.

[30] Computer Science Teachers Association CSTA, "About the CSTA K-12 Computer Science Standards", Retrieved from: https://www.csteachers.org/page/standards, Accessed 5-29-2020.

[31] Brennan, K., & Resnick, M., "New frameworks for studying and assessing the development of computational thinking", In Proceedings of the 2012 Annual Meeting of the American Educational Research Association, pp. 1-25, 2012.

[32] Isbell, C. L., Stein, L. A., Cutler, R., Forbes, J., Fraser, L., Impagliazzo, J., ... & Xu, Y., "(Re) defining computing curricula by (re) defining computing", ACM SIGCSE Bulletin, 41(4), pp. 195-207, 2010.

[33] College Board, "AP Computer Science Principles", Retrieved from: https://apstudent.collegeboard.org/apcourse/ap-computer-science-principles/course-details, Accessed 5-29-2020.

[34] College Board, "AP Computer Science Principles, Including the Curriculum Framework", 2017.

[35] College Board, "AP Computer Science Principles: The Exam", https://apcentral.collegeboard.org/courses/ap-computer-science-principles/exam, Accessed 5-29-2020.

[36] College Board, "AP Students: Exam Fees", https://apstudents.collegeboard.org/exam-policies-guidelines/exam-fees, Accessed 5-29-2020.

[37] ABET, "Criteria for Accrediting Engineering Programs, 2020-2021", https://www.abet.org/accreditation/accreditation-criteria/criteria-for-accrediting-engineering-programs-2020-2021/, Accessed 5-29-2020.

[38] Cohen, J., "Statistical Power Analysis for the Behavioral Sciences", Lawrence Erlbaum, Hillsdale, NJ, 1988.

[39] Mendoza Diaz, N., "Hispanics in Engineering", Proceedings of the 121st ASEE Annual Conference and Exposition, June 2014, Indianapolis, IN, USA, 2014.