Engineering Reimagined: (Re)designing Next-Generation Engineering Curricula for Industry 5.0

Abstract

This paper presents a framework for redesigning engineering curricula to meet Industry 5.0 demands and describes the reimagining process. While all our engineering programs executed the process, this paper highlights the redesign of our computer engineering curriculum using the framework for illustrative purposes. The process commences with program faculty examining the latest curriculum via a set of guided questions to identify if and how it meets the future needs of industry and the learning approaches of our students, present and future. In the computer engineering case, the program faculty identified that the curriculum was lacking in some areas and could be reimagined such that it can: provide skills that can be adjusted and adapted to new areas, allow for more flexibility and humanity in the treatment of students and faculty, focus on future fields (e.g., artificial intelligence, machine learning, internet of things) while meeting core learning outcomes, firmly push students towards independent learning, and provide the big picture of the learning outcomes and trajectory early and often. In phase two of the process, the faculty reached out to industry partners to obtain insight into the desired skills of the computer engineering graduates of tomorrow, our learning and teaching support staff to reveal modern teaching practices and tools that our next generation curricula could leverage, and other computer engineering programs to identify how they are adapting to the same challenges. The findings from the research, detailed in the remainder of this paper, were used to fuel the third phase of *Engineering Reimagined.* The program faculty holistically considered all the feedback, including that of other university-wide committees. They focused on inclusive excellence and student retention to define the learning outcomes of the entire program and map them in knowledge areas, which are then encapsulated in classes (new and existing) that are finally scaffolded in our next-generation computer engineering curriculum.

Introduction

More than ten years have passed since our institution, like many others worldwide, has addressed the challenges of Industry 4.0 and (re)imagined and (re)designed our engineering curricula [1]. Since then, innovations such as distributed computing and robotics, the Internet of things, multi-agent systems, semantic web, complex adaptive systems, artificial intelligence and machine learning, and self-organizing business processes; have emerged and accelerated the transition from Industry 4.0 to Industry 5.0 [2]. The overarching problem is that the industry careers presented to our current graduates have changed fundamentally [3], and our present curricula are increasingly becoming less fit to prepare our students to fill the needs of their prospective employers. Also, our instructional methodologies are rapidly becoming obsolete because they are disconnected from the needs of Industry 5.0 and, more importantly, are not tailored to capture the imagination and attention of our students, who desire curricula that efficiently and effectively educate them in professional competencies [5]. Our motivation to introduce the framework presented in this paper is to aid educators in rethinking and redesigning engineering curricula such that our future students can be better prepared to transcend the competency gap between Industry 4.0 and 5.0 [6].

The U.S. workforce faces a crisis today, where women and minorities are increasingly underrepresented in science, technology, engineering, and math (STEM) fields [7]. This is at a time when growth in STEM jobs has outpaced growth in all other occupations. U.S. companies have resorted to importing STEM workers to fill this need and remain competitive. However, for the U.S. to lead the technological innovation in Industry 5.0, the education sector must be restructured to focus on increasing the participation of underrepresented groups in STEM majors and supporting this demographic to ensure retention and success in the classroom and the workforce [8]. Considering our evolving demographics, supporting the student success of underrepresented minorities in STEM fields is of significant importance for the survival of educational institutions [9]. As educators, we must strive to be forward-looking and create curricula with inclusive excellence in mind, nurturing the identities of our students to create opportunities for cross-cultural learning [10], a significant driver for creativity and innovation.

The landscape of our digital world has recently shifted focus towards data science, cloud computing, artificial intelligence (AI), and cybersecurity [11]. Even accrediting agencies, such as ABET, are considering initiatives to support adding a data science requirement to all engineering programs [12]. These technological innovations are fundamentally pushing the industry to the next mode of operation (Industry 5.0), rewriting the script on essential occupational skills for the jobs of tomorrow. STEM competencies are and will be increasingly desired by industry in non-STEM fields, creating opportunities for a more mobile workforce. Companies say that the lines are blurring between industry and academia and that we as educators need to make deeper connections and a commitment to integration [13]. While the boundary between disciplines still exists, they tend to overlap more when it comes to the workplace.

Our goal as engineering educators should be to balance the breadth of cross-disciplinary knowledge and practices in next-generation engineering curricula while integrating interdisciplinary learning and discipline-specific competencies [14]. We need to educate students to solve problems in a cross-disciplinary environment. Thus, modern curricula must focus more on cultivating diverse team working skills while nurturing individual values, knowledge, and skills [15]. Critical thinking plays an ever-increasing important role in the success of our students in their future careers with blurred lines in diverse teams, and numerous studies show that it is a skill that cannot be taught, rather developed through experiential learning in a curriculum that vertically integrates problem-solving [16].

Another challenge introduced by Industry 5.0 is that instead of designing a full discipline-specific product, engineers are increasingly acting as system integrators; therefore, engineers must possess the ability to quickly learn and adapt material from other STEM and non-STEM fields. Industry 5.0 projects are complex, multidimensional, and fast-evolving, requiring critical thinking and problem-solving abilities from engineers beyond their STEM foundation. The engineers of tomorrow must have the ability to communicate and work in multi- or inter-disciplinary teams. Furthermore, these engineers must be innovative and creative and, most importantly, open to ideas from others. Simply put, educating engineers in the technical domain is not enough [17].

This paper presents an *Engineering Reimagined* framework and processes that can aid engineering educators in their curriculum (re)design to meet the many outlined challenges of Industry 5.0 and beyond. It consists of three phases, described in the next section, and executed, for illustrative purposes, by our computer engineering program faculty, with results presented in the following sections.

Framework and Methods

The *Engineering Reimagined* process commences with program faculty examining the latest curriculum via a set of guided 68 questions (provided in Appendix A) to identify if and how it meets the future needs of industry and the learning approaches of our students, present and future. The questions are meant to guide a more profound discussion and introspection of the current state of the curriculum. In the computer engineering case at our institution, the faculty met for a total of 8.5 hours in 7 meetings in June and July of 2021 and identified that the program was lacking in some areas and could be reimagined such that it can: provide skills that can be adjusted and adapted to new areas, allow for more flexibility and humanity in the treatment of students and faculty, focus on future fields (e.g., artificial intelligence, machine learning, internet of things) while meeting core learning outcomes, strongly push students towards independent learning, and provide the big picture of the learning outcomes and trajectory early and often.

With a deeper understating of the state of affairs by the program faculty collective illuminated by phase one of *Engineering Reimagined*; in phase two, the program faculty are encouraged to reach out to industry partners to obtain insight into the desired skills of the engineering graduates of tomorrow; learning and teaching support staff at their institution and beyond to reveal modern teaching practices and tools that could be leveraged by next-generation curricula; and investigate other engineering programs to identify how they are adapting to the challenges of meeting Industry 5.0 demands.

The findings from the research (illustrated through a case study of the execution of the framework on our computer engineering curriculum) in the remainder of this paper are then used to fuel the third phase of *Engineering Reimagined*, where the program faculty holistically consider all the feedback, including that of other university-wide committees, focus on inclusive excellence and student retention, to define the learning outcomes of the entire program and map them in knowledge areas, which are then encapsulated in classes (new and existing) that are finally scaffolded in a next-generation engineering curriculum.

Research of Other Institutions' Engineering Curricula

The primary objective of a Research of Other Institutions (ROI) task force (TF) is to identify and research at least five top-ranked programs from regional, national, and top-tier institutions. The ROI TF at our institution identified and studied 12 programs of B.S. in Computer Engineering (BSCO) from nationally top-ranked institutions, five of which were from ones not offering doctorate degrees, two from most innovative engineering schools in the north also not offering doctorate degrees, and five from regional universities offering doctorate degrees. The TF deep-

scanned the 12 BSCO curricula, including topics covered, instructional methodologies, etc., aiming to reveal innovative curriculum approaches.

The TF first examined the U.S. News Best Colleges for Computer Engineering (Doctorate not Offered) [18], which listed 16 national universities. Several of the institutions in this list were not considered because they did not have a program explicitly titled BSCO, such as Olin Institute of Technology (Olin), which has a hybrid "B.S. in Electrical and Computer Engineering" program. Furthermore, 6 out of the 16 ranked institutions were in the California State University system; thus, only 1 out of them, the number 2 ranked California Polytechnic State University-San Luis Obispo (Cal Poly), was selected as the best representative of the group. The TF excluded the military academies - Airforce, Naval, and Military - because of the differences in our student populations. Of the remaining institutions, the following were selected for further research: Rose-Hulman Institute of Technology (RHIT), Cal Poly, Milwaukee School of Engineering (MSOE), Bucknell University (Bucknell), and Embry-Riddle Aeronautical University – Prescott (ERAU).

The TF further focused on the U.S. News rankings of Most Innovative Schools in Engineering – Regional Universities North [19], and from the 21 listed institutions, identified two that offered a BSCO program and were in the "Doctorate not Offered" category: Merrimack College (Merrimack), and The College of New Jersey (TCNJ). Finally, the TF expanded the search to obtain insight from regional universities that offer doctorate degrees and identified the following institutions of interest: Northeastern University (NEU), Boston University (BU), University of New Hampshire (UNH), University of Rhode Island (URI), Tufts University (Tufts).

The TF then captured all the BSCO curricula tracking sheets from the 12 identified institutions and researched their structure and course offerings. The findings of the TF are the following and illustrated in Figure 1.

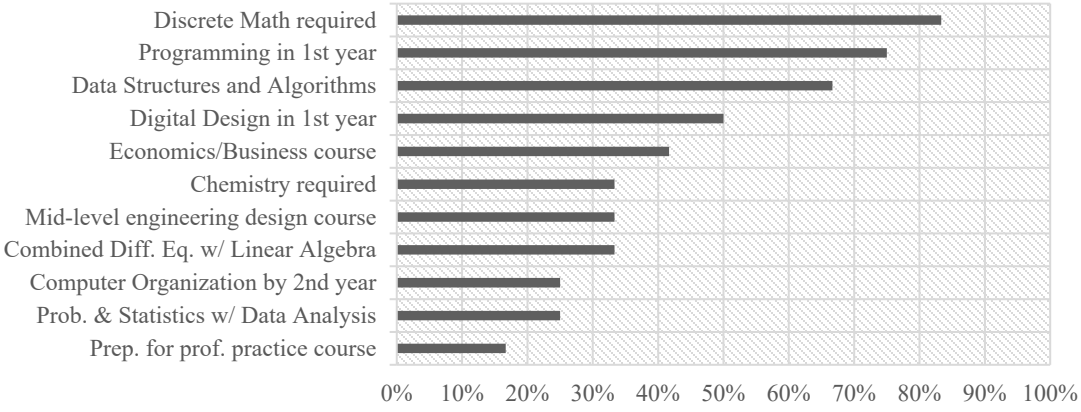

Figure 1. ROI TF findings of innovative BSCO curricula trends.

Of the 12 programs studied, 9 require a programming course in year 1 of their BSCO curriculum. 8 Programs offer data structures and algorithm courses, either combined or two separate courses. Half of the programs require *Digital Design* in year 1 of the curriculum. An *Economics* or *Business* course is required in 5 programs; some count it as a required Humanities and Social

Science (HUSS) course. A third of the programs: combine *Differential Equations* with *Linear Algebra* in a single course, have a required mid-level (sophomore or junior) engineering design course, and require *Chemistry* for BSCO majors. A quarter of the analyzed programs: require a combined course in *Probability and Statistics* with *Data Analysis/Science* and require a course in *Computer Organization* (whether an entire course or a part of a class) by the sophomore year in the curriculum. Only two programs have no explicit *Discrete Math* course and require preparation for professional practice (co-op/career prep/internship) course. Some programs such as NU's and BU's BSCO place little emphasis on electrical engineering while placing a lot of emphasis on applied BSCO courses. Merrimack requires the least math (no *Multivariable Calculus* and *Differential Equations*) and offers a discrete version of *Signals and Systems.*

Industry Feedback

The primary objective of a Research of Industry Feedback (IF) TF in the *Engineering Reimagined* process is to explore the necessary competencies of future engineering graduates as dictated by industrial partners and employers of the program's graduates. The IF TF at our institution generated a survey to get feedback from the industry, with questions in the study grouped into four categories: BSCO knowledge areas, soft skills, programming languages, and recommended courses/topics.

1. Computer Engineering Knowledge Areas
In this category, our survey asked respondents to rank the five most important computer engineering knowledge areas (out of the 12) to their companies and teams. The list of all 12 computer engineering knowledge areas, shown in Table 1, are detailed in the IEEE/ACM Curriculum Guidelines for Undergraduate Degree Programs in Computer Engineering [20] and are broken down 135 knowledge units, further divided into a total of 908 learning outcomes. This question helped the TF define important knowledge that students should master before graduating in order to prepare them for their future careers in Industry 5.0.

At the time of this writing, we received seven responses from our survey, a very small number that is hard to draw a lot of firm conclusions from. Nonetheless, we plan to expand the number of respondents to be statistically more significant and include a larger variety of companies and representative positions within the same. The responses are from engineers from several companies, including *Apple*, *The MathWorks*, *Verizon*, *Toast*, *Randstad Technologies*, and *Altaeros*. The positions of our respondents are Salesforce QA Engineer II, RF Systems Engineer, Senior Software Engineer, Electrical Engineer, Senior Electrical Engineer, and Solutions Architect. The results of our survey are shown in Table 1.

According to the results, most companies desire well-trained BSCO graduates in circuits and electronics, computer networks, and software design. Surprisingly, contrary to faculty belief, not many companies deemed *Computer Architecture and Organization* a top skill of prospective employees.

Table 1. Industry feedback ranking desired knowledge areas in computer engineering.

Rank of Importance (5 most, 1 least)	5	4	3	2	1	Weighted Avg
Circuits and Electronics (CAE)	1	2	1			**2.29**
Computer Networks (NWK)	2		1	1	1	**2.29**
Software Design (SWD)	2		1		2	**2.14**
Embedded Systems (ESY)	1			2	1	**1.43**
Systems and Project Engineering (SPE)	1	1				**1.29**
Preparation for Professional Practice (PPP)		1		2		**1.14**
Signal Processing (SGP)		1	1		1	**1.14**
Computing Algorithms (CAL)		1	1			**1.00**
Digital Design (DIG)			1	1	1	**0.86**
Information Security (SEC)		1		1		**0.86**
System Resource Management (SRM)			1			**0.43**
Computer Architecture and Organization (CAO)					1	**0.14**

2. Soft Skills

In this category, our IF TF designed a survey asking respondents' opinions on the five most important soft skills that they are looking for in their future team members. The list of essential BSCO soft skills in this survey are *Presentation*, *Problem Solving*, *Teamwork*, *Leadership*, *Communication*, *Interpersonal*, *Flexibility*, *Decision Making*, *Open Mindedness*, *Self Learner*, *Creativity*, and others [21]. According to our survey, the most critical soft skills that employers are looking for in their candidates are *Problem Solving* and *Self Learner*. *Communication* is another essential soft skill that we should prepare for our future graduates. Figure 2 presents our survey results, which surprisingly reveal little industry desire for leadership and flexibility skills from our prospective graduates.

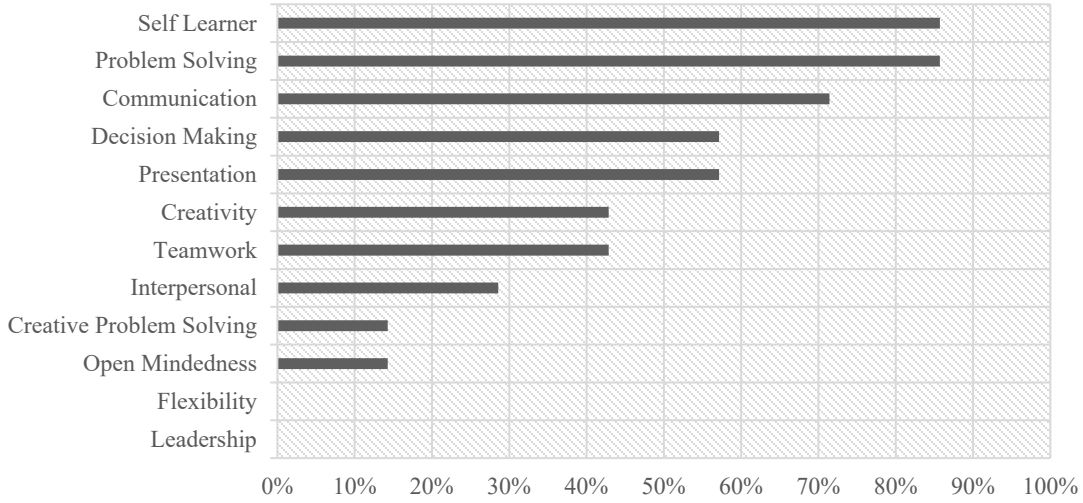

Figure 2. Industry feedback on soft skills required from prospective graduates.

3. Programming Languages
In this category, this survey asks respondents' opinions on the five most important programming languages for their future employees. This survey's list of programming languages included C/C++, Python, Verilog/VHDL, HTML, JavaScript, CSS, SPICE, and others. According to the results shown in Figure 3, C/C++ and Python are the most sought-after programming languages that our future graduates should have a good command of.

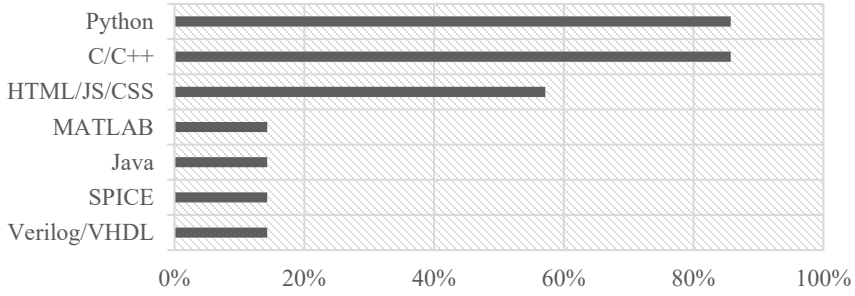

Figure 3. Industry feedback on desired knowledge of programming languages.

4. Recommended Courses and Topics
In this category, our survey asked respondents if they have any recommended courses or topics that our BSCO program should focus on. Our respondents suggested the following topics that we should include in our next-generation curriculum to better prepare our future graduates for Industry 5.0: software architecture, JavaScript, Java, C/C++, wireless and wired network standards, artificial intelligence, basic knowledge of other engineering disciplines, and focus on soft skills and emotional intelligence.

Instructional Technologies

The Teaching and Learning Collaborative (TLC) outreach TF objective was to connect with instructional design professionals, within and/or outside the institution, to reveal modern teaching practices and tools that next-generation engineering curricula could leverage. As stated before, the problem with old-age instructional methodologies is that they are not tailored to capture the imagination and attention of our present-day and future students, who increasingly desire curricula (and in turn classroom instruction) that can efficiently and effectively train them in professional competencies sought after by Industry 5.0.

Our BSCO TLC TF met on two occasions with the instructional design staff at our institution. Regarding reimagining the curricula, some of the available resources such as engaging the instructional designers will have to be methodically utilized over time to create opportunities for our students that will include active, project-based, and collaborative learning; with the aim to increase involvement and retention, especially in first-year engineering courses.

It was identified that combined use of synchronous and asynchronous learning tools might be the most effective strategy for effective teaching and learning, such as incorporating some (or all) of these: video micro-lectures, flipped classrooms, social reading with the synthesis of knowledge

via summaries, intra-classroom group work via think-pair-share and extra-classroom activities organized via Brightspace, and multiple levels of try-fail-iterate assessments.

Engineering Reimagined Outcome Recommendations

The ROI TF recommends introducing programming in the first year of the BSCO curriculum and introducing additional programming courses later. We recommend separating our *Microcontrollers using C Programming* course into two courses, one being *C Programming for Engineers*, potentially taught in the first year of the curriculum, and the other, *Microcontrollers*, to be offered at the same level as the current combined course, where more time can be devoted to projects and problem-solving. We recommend moving the *Object-oriented Programming for Engineers* course in year two and restructuring it to focus solely on programming and problem-solving using the object-oriented methodology; the current focus on data structures in this course should be moved to a new course titled *Data Structures and Algorithms*, focusing on algorithm complexity analysis and design, as well as Linked Lists, Trees, Queues, Hashes, Graphs, etc., and using them for problem-solving. The TF recommends that the current *Applied Programming Concepts* course remains or be rebranded as *Software Engineering* to align with other curricula.

The ROI TF recommends streamlining the math courses in the BSCO curriculum. We need to investigate whether we need Multivariable Calculus, as many other curricula do not include it. Our current *Discrete Math* course would need to be restructured (and rebranded to *Discrete Structures*) to be less focused on Boolean Algebra, which is already covered in our *Digital Logic* class, and more so on the following topics: Sets, Set Operations, Counting, Combinatorics, Pigeonhole Principle, Modular Arithmetic, Divisibility and Encryption, Mathematical Induction, Asymptotic Time Complexity, Search and Sort, Recurrence Relations, and Graph Theory. The TF also recommends that we offer a new version of *Probability and Statistics with Data Analytics/Visualization* much earlier in the curriculum (not in year 4) to be used and applied in other core courses. Many curricula combine *Differential Equations with Linear Algebra*; thus, the TF recommends we do the same. Finally, the TF recommends investigating the possibility of offering a *Discrete Signals and Systems* version for BSCO majors.

For Humanities and Social Sciences (HUSS), the ROI TF recommends adding a 0-credit hour *Co-op Preparation* course to prepare students uniformly better for co-ops and careers. Most universities require an *Ethics* course (*Ethics*, *Ethics for Professional Managers and Engineers*, *Technology as a Service to Humanity*, etc.), we should therefore consider the same. The TF finally recommends making two of the HUSS courses chosen by the BSCO Committee and one free elective chosen by the student.

The IF TF recommends, according to the results of our survey, for our BSCO program to focus on improving students' problem-solving skills and self-learner skills because these skills will be particularly important in their future careers. Our students should be mastered in C/C++ and Python because these programming languages have been used in various teams. Moreover, we should find ways to help our future graduates prepare for their future careers. Currently, the preparedness of our BSCO graduates is on average compared to other schools. One suggestion

that we have is to expose students to real-world experiences, such as having a yearly project, where students must use knowledge from various classes and work on a project at the end of each year. This way, students could improve their problem-solving and self-learner skills, and it will help them prepare for their future careers in Industry 5.0.

The TLC TF recommendations included:

- Instructional Design with the help of professional staff
- Uniform use of a Learning Management System (LMS), such as Brightspace
- Gradescope with automated grading of programming assignments
- Integrating multidisciplinary project-based learning
- Flipped classroom model
- Micro-lectures, using Panopto
- Social reading (AKA social annotation) tools such as: perusall and hypothes.is
- Synthesis of knowledge via summaries
- Group work (think-pair-share) and use of LMS groups for discussions
- Peer-to-peer feedback process
- Multiple levels of try-fail-iterate assessments: students first grade themselves, then work in groups to get peer feedback, then the instructor assesses knowledge
- Asynchronous extracurricular team activities enforced with assignments and concrete deliverables
- Meta-cognitive scaffolding of the curriculum. Faculty will need to explicitly show and tell students the big picture of how the class learning outcomes map to the overall program outcomes
- Integrate instructional designers across first year engineering courses
- Look at classes that students are excited about and talk to instructors to see what they did and emulate that behavior
- Embedded tutors and teaching assistants to help retention.

The TLC TF highly recommends the use of embedded tutors, especially in 'gateway' courses (2nd-year students, first core courses in their major); and that each class should present meta-cognitive scaffolding of the curriculum, where faculty will need to explicitly show and tell students the big picture of how the class learning outcomes map to the overall program outcomes. The TLC TF will involve the TLC instructional designers in all future *Engineering Reimagined* discussions so that they are cognizant of our efforts and can fully participate in the process.

Conclusions

The third and final phase of *Engineering Reimagined* is to consider all the obtained feedback and map learning outcomes to courses and scaffold the next-generation curriculum. To further aid us in this effort, our BSCO faculty at our institution executed a self-audit based on ACM/IEEE computing curricula recommendations, specifically on the latest Curriculum Guidelines for Undergraduate Degree Programs in Computer Engineering (CE2016) [2], to quantify to what degree we meet the standards. The CE2016 guidelines divide BSCO curricula into 12

Knowledge Areas (KA) (Circuits and Electronics, Computing Algorithms, Computer Architecture and Organization, Digital Design, Embedded Systems, Computer Networks, Preparation for Professional Practice, Information Security, Signal Processing, Systems and Project Engineering, Systems Resource Management, and Software Design) composed of a total of 135 Knowledge Units (KU), further detailed by a total of 908 Learning Outcomes (LO). The program faculty cross-examined the BSCO curriculum and identified if and where the LOs were covered. The self-audit gave us a better understanding of our current curriculum, revealed deficiencies and excess, and informed our near- and long-term faculty hiring process of needed expertise.

Furthermore, the BSCO program faculty examined the ACM/IEEE Computing Curricula 2020 Paradigms for Global Computing Education (CC2020) [22], which focuses on the core competencies of graduates from next-generation computing programs. CC2020 details 34 competencies divided into six knowledge areas (Hardware, Systems Architecture and Infrastructure, Software Fundamentals, Software Development, Systems Modeling, and Systems and Organizations). In the last phase of *Engineering Reimagined,* the faculty will need to identify how the CE2016 LO map to the CC2020 competencies, such that our reimagined BSCO curriculum can transcend the competency gap between the graduates of today and the careers of tomorrow.

The findings from the research obtained by executing the process based on the presented framework provide vital data for the third phase of *Engineering Reimagined*; where the program faculty should holistically consider all the feedback and focus on inclusive excellence and student retention to define the learning outcomes of the entire program and map them in knowledge areas, which are then to be encapsulated in classes that are scaffolded into a next-generation engineering curriculum adapted to the challenges and demands of Industry 5.0.

References

[1] W. Maisiri and L. van Dyk, "Industry 4.0 Competence Maturity Model Design Requirements: A Systematic Mapping Review," in *2020 IFEES World Engineering Education Forum - Global Engineering Deans Council (WEEF-GEDC)*, Nov. 2020, pp. 1–6. doi: 10.1109/WEEF-GEDC49885.2020.9293654.

[2] X. Xu, Y. Lu, B. Vogel-Heuser, and L. Wang, "Industry 4.0 and Industry 5.0—Inception, conception and perception," *J. Manuf. Syst.*, vol. 61, pp. 530–535, 2021.

[3] P. O. Skobelev and S. Y. Borovik, "On the way from Industry 4.0 to Industry 5.0: From digital manufacturing to digital society," *Ind. 40*, vol. 2, no. 6, pp. 307–311, 2017.

[4] P. K. R. Maddikunta *et al.*, "Industry 5.0: A survey on enabling technologies and potential applications," *J. Ind. Inf. Integr.*, p. 100257, 2021.

[5] F. Sánchez Carracedo *et al.*, "Competency Maps: An Effective Model to Integrate Professional Competencies Across a STEM Curriculum," *J. Sci. Educ. Technol.*, vol. 27, no. 5, pp. 448–468, Oct. 2018, doi: 10.1007/s10956-018-9735-3.

[6] D. Gürdür Broo, O. Kaynak, and S. M. Sait, "Rethinking engineering education at the age of industry 5.0," *J. Ind. Inf. Integr.*, vol. 25, p. 100311, Jan. 2022, doi: 10.1016/j.jii.2021.100311.

[7] R. Varma, "US science and engineering workforce: Underrepresentation of women and minorities," *Am. Behav. Sci.*, vol. 62, no. 5, pp. 692–697, 2018.

[8] C. T. Belser, M. Shillingford, A. P. Daire, D. J. Prescod, and M. A. Dagley, "Factors influencing undergraduate student retention in STEM majors: Career development, math ability, and demographics.," *Prof. Couns.*, vol. 8, no. 3, pp. 262–276, 2018.

[9] A. M. Ogilvie and D. B. Knight, "Engineering transfer students' reasons for starting at another institution and variation across subpopulations," *J. Hisp. High. Educ.*, vol. 19, no. 1, pp. 69–83, 2020.

[10] A. Godwin and A. Kirn, "Identity-based motivation: Connections between first-year students' engineering role identities and future-time perspectives," *J. Eng. Educ.*, vol. 109, no. 3, pp. 362–383, 2020.

[11] S. Liu, J.-L. Gaudiot, and H. Kasahara, "Engineering education in the age of autonomous machines," *ArXiv Prepr. ArXiv210207900*, 2021.

[12] J. R. Blair, L. Jones, P. Leidig, S. Murray, R. K. Raj, and C. J. Romanowski, "Establishing ABET accreditation criteria for data science," in *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*, 2021, pp. 535–540.

[13] S. Ivey, "Inspiring the next generation mobility workforce through innovative industry–academia partnerships," in *Empowering the New Mobility Workforce*, Elsevier, 2019, pp. 317–348.

[14] E. Kähkönen and K. Hölttä-Otto, "From crossing chromosomes to crossing curricula–a biomimetic analogy for cross-disciplinary engineering curriculum planning," *Eur. J. Eng. Educ.*, pp. 1–19, 2021.

[15] P. S. Nurius and S. P. Kemp, "Individual-level competencies for team collaboration with cross-disciplinary researchers and stakeholders," in *Strategies for Team Science Success*, Springer, 2019, pp. 171–187.

[16] H. J. Yazici, L. A. Zidek, and H. St Hill, "A study of critical thinking and cross-disciplinary teamwork in engineering education," in *Women in industrial and systems engineering*, Springer, 2020, pp. 185–196.

[17] J. Qadir, K.-L. A. Yau, M. Ali Imran, and A. Al-Fuqaha, "Engineering Education, Moving into 2020s: Essential Competencies for Effective 21st Century Electrical and Computer Engineers," in *2020 IEEE Frontiers in Education Conference (FIE)*, Oct. 2020, pp. 1–9. doi: 10.1109/FIE44824.2020.9274067.

[18] "The Best Colleges for Computer Engineering," *US News & World Report*. https://www.usnews.com/best-colleges/rankings/engineering-computer (accessed Oct. 29, 2021).

[19] "The Most Innovative Regional Universities in the North," *US News & World Report*. https://www.usnews.com/best-colleges/rankings/regional-universities-north/innovative?study=Engineering (accessed Oct. 29, 2021).

[20] "ce2016-final-report.pdf." Accessed: Nov. 14, 2021. [Online]. Available: https://www.acm.org/binaries/content/assets/education/ce2016-final-report.pdf

[21] M. Schipper and E. van der Stappen, "Motivation and attitude of computer engineering students toward soft skills," in *2018 IEEE Global Engineering Education Conference (EDUCON)*, Apr. 2018, pp. 217–222. doi: 10.1109/EDUCON.2018.8363231.

[22] CC2020 Task Force, *Computing Curricula 2020: Paradigms for Global Computing Education*. New York, NY, USA: ACM, 2020. doi: 10.1145/3467967.

Appendix A: Engineering Reimagined Questions

1. Does your program focus on the acquisition and development of fundamental knowledge in engineering?
2. Does your program allow students to learn, analyze, intellectualize, design, develop, and test complex systems?
3. Does your program teach students to use the right side of the brain?
4. Does your program teach students of product/service/system life cycles?
5. Does your program provide students with experience in system thinking?
6. Does your program teach students to accept failure and take risks?
7. Does your program teach students that problems do not have the right or correct answers but best answers given a set of constraints?
8. Does your program teach students when *good* is good enough?
9. Does your program teach students the concept of value creation and the dangers of over-engineering a design?
10. Does your program teach students about design consistency?
11. Does your program teach students to think about the big picture when it comes to design?
12. Does your program teach students about the Agile methodology, such as rapid iteration, rapid prototyping, and continuous customer involvement?
13. Does your program have professors playing ignorant and providing problems with incomplete information or models so that students have to make assumptions?
14. Does your program have professors playing ignorant and providing problems with too much information so that students have to makes sense of every aspect of the problem?
15. Does your program make students deal with the uncertainty of real problems?
16. Does your program teach students how to think (in and out of the box)?
17. Does your program teach students how to learn?
18. Does your program teach students how to ask the right questions?
19. Does your program teach students that there are no stupid questions?
20. Does your program train students how to distill answers to questions that can be fed into a computer?
21. Does your program teach students how to define a problem?
22. Does your program teach students how to acquire and interpret knowledge from the Internet?
23. Does your program teach students to be independent, critical thinkers?
24. Does your program teach students to be data literate?
25. Does your program teach students about big data, data analytics, cybersecurity, or cloud computing?
26. Does your program teach students to have algorithmic thinking and programming skills?
27. Does your program teach students how to visualize complex or rich data sets?
28. Does your program teach students about social and human factors?
29. Does your program teach students to be creative and address different ideas?
30. Does your program teach students about the entrepreneurial mindset?
31. Does your program teach students how to think divergently?
32. Does your program educate students on designing within an already existing framework (i.e., a previous solution exists)?
33. Does your program educate students on how to work in multi- or inter-disciplinary teams?

34. Does your program teach students how to collaborate with people outside of the discipline of engineering?
35. Does your program teach students how to respect other people's views?
36. Does your program teach students leadership (teamwork, creativity, etc.) skills?
37. Does your program teach the business side of engineering?
38. Does your program do synthesis within the discipline, within engineering, and outside of engineering?
39. Does your program teach when to incorporate social elements into system analysis and design?
40. Does your program teach students to collaborate with stakeholders outside of their domain?
41. Does your program teach students how to use people outside of the domain to reduce the number of unknown unknowns?
42. Does your program help students think about their lives, career goals, and plans to lead a productive and meaningful life?
43. Does your program teach students how to communicate in both face-to-face interactions or via screens?
44. Does your program teach students how to listen?
45. Does your program teach students how to visualize (sketch and diagram) solutions?
46. Does your program teach students how to write a professional email?
47. Does your program teach students how to communicate with people from different backgrounds?
48. Does your program teach students about the challenges of dealing with people who do not believe in science, have misconceptions, or have biases?
49. Does your program teach students that people from different backgrounds could interpret data in other ways?
50. Does your program teach students the power of persuasion?
51. Does your program teach students to ability to do an elevator pitch?
52. Does your program teach students to be more comfortable with words than numbers?
53. Does your program teach students about public policy and regulations?
54. Does your program teach students about societal issues?
55. Does your program teach students about the grand challenges?
56. Does your program teach students how to treat others with respect and dignity?
57. Does your program teach students how to become empathetic engineers?
58. Does your program teach students diversity in thought?
59. Does your program teach students about working with people of different backgrounds and working with culturally diverse contexts?
60. Does your program teach students about intercultural collaboration and globalization?
61. Does your program teach students about ethical issues arising from cultural differences?
62. Does your program make students comfortable with the uncomfortable?
63. Does your program make students the primary player in their learning process?
64. Does your program incentivize feedback on progress/growth vs. feedback on achievement/knowledge?
65. Does your program teach students the basics of project management?
66. Does your program include the right blend of theory and practicality?
67. Does your program include the right blend of teaching for life and teaching for the job?
68. Is your program adaptable?