# AC 2011-1050: COMPUTATIONAL EXPERTISE IN ENGINEERING: ALIGNING WORKFORCE COMPUTING NEEDS WITH COMPUTER SCIENCE CONCEPTS.

**Claudia Elena Vergara, Michigan State University**

Claudia Elena Vergara. PhD Purdue University. Fields of expertise: Plant Biology and STEM Education Research. Dr. Vergara is a Postdoctoral Fellow at the Center for Engineering Education Research (CEER) at Michigan State University. Her research interest is in STEM education through research projects on instructional design, implementation and assessment of student learning, aimed to improve science, engineering and technology education.

**Mark Urban-Lurain, Michigan State University**

Director of Instructional Technology Research & Development Division of Science and Mathematics Education College of Natural Science Michigan State University

**Abdol-Hossein Esfahanian, Michigan State University**

Abdol-Hossein Esfahanian received his B.S. degree in Electrical Engineering and the M.S. degree in Computer, Information, and Control Engineering from the University of Michigan in 1975 and 1977 respectively, and the Ph.D. degree in Computer Science from Northwestern University in 1983. He was an Assistant Professor of Computer Science at Michigan State University from September 1983 to May 1990. Since June 1990, he has been an Associate Professor with the same department, and from August 1994 to May 2004, he was the Graduate Program Director. He has developed a number of software packages which have been used both inside and outside the University. He was awarded 'The 1998 Withrow Exceptional Service Award', and 'The 2005 Withrow Teaching Excellence Award'. Dr. Esfahanian has published articles in journals such as IEEE Transactions, NETWORKS, Discrete Applied Mathematic, Graph Theory, and Parallel and Distributed Computing. He was an Associate Editor of NETWORKS, from 1996 to 1999. He has been conducting research in applied graph theory, computer communications, fault-tolerant computing, Information Technology, and databases.

**Daina Briedis, Michigan State University**

DAINA BRIEDIS is a faculty member in the Department of Chemical Engineering and Materials Science at Michigan State University. Dr. Briedis has been involved in several areas of education research including student retention, curriculum redesign, and the use of technology in the classroom. She is a co-PI on two NSF grants in the areas of integration of computation in engineering curricula and in developing comprehensive strategies to retain early engineering students. She is active nationally and internationally in engineering accreditation and is a Fellow of ABET.

**Dr. Neeraj Buch, Michigan State University**
**Thomas F. Wolff, Michigan State University**

Dr. Thomas F. Wolff is Associate Dean of Engineering for Undergraduate Studies at Michigan State University. In this capacity, he is responsible for all activities related to student services (academic administration, advising, career planning, women and diversity programs, etc.) and curricular issues. He is principal investigator on several NSF grants related to retention of engineering students. As a faculty member in civil engineering, he co-teaches a large introductory course in civil engineering. His research and consulting activities have focused on the safety and reliability of hydraulic structures, and he has participated as an expert in three different capacities regarding reviews of levee performance in Hurricane Katrina. He is a three-time recipient of his college's Withrow Award for Teaching Excellence, a recipient of the Chi Epsilon Regional Teaching Award, and a recipient of the U.S. Army Commander's Award medal for Public Service. In 2010, he was elected to the National Council of Chi Epsilon, the civil engineering honor society, and serves as National Marshal of that organization.

**Jon Sticklen, Michigan State University**

Jon Sticklen is the Director of the Center for Engineering Education Research at Michigan State University. Dr. Sticklen is also Director of Applied Engineering Sciences, an undergraduate bachelor of science degree program in the MSU College of Engineering. He also is an Associate Professor in the Department of Computer Science and Engineering. Dr. Sticklen has lead a laboratory in knowledge-based systems focused on task specific approaches to problem solving. Over the last decade, Dr. Sticklen has pursued engineering education research focused on early engineering; his current research is supported by NSF/DUE and NSF/CISE.

**Ms. Cindee Dresen**
**Kysha L. Frazier, Corporation for a Skilled Workforce**
**Louise Paquette, Lansing Community College**

Louise A. Paquette Lansing Community College Mathematics and Computer Science Department

Degrees EdS in Curriculum and Instruction with a minor emphasis in Systems Science, Michigan State University, 1982 MAT in Mathematics Education with a minor emphasis in Computer Science, Michigan State University, 1978 BA in Mathematics Education with a minor in business, Michigan State University, 1969

Professional Experience Sum 1983 present Mathematics professor at Lansing Community College (full-time since January 2000) Sept 1994 present Coordinator of the 2+2+2 Engineering Program. Responsibilities include: arranging tours, orientations, and meetings; academic advising; mentoring; monitoring progress; tutoring. Jan 97 Jun 09 Coordinator of the Liberal Studies Divisional Awards Jan 95 present Title III Academic Advisor Fall 96,97,98,99 Professor at Lyman Briggs College, Michigan State University Spring 94 Visiting Instructor, Mathematics Department, Michigan State University Sept 82 Dec 85 Assistant Instructor, Mathematics Department, Michigan State University Sept 76 Jun 82 Graduate Assistant, Mathematics Department, Michigan State University Sept 69 Jun 75 High school mathematics/computer science teacher for L'Anse Creuse Public Schools, Mt Clemens, MI 48043

Presentations Fall1996 I gave brief demonstrations of the capabilities of the TI-092 graphing calculator to mathematics faculty. I continued to meet with some faculty throughout Spring 1997 to continue the discussion about the calculator. July1995 I gave a presentation "Use of the Graphing Calculator in the Classroom" at the Liberal Studies Division Sharing Meeting at LCC. I also conducted a Professional Development Workshop for the LCC Science Department on use of the TI-82 graphing calculator. Sept1994 I conducted a Professional Development Meeting for the LCC Mathematics Department on the features of the TI-82 graphing calculator. Mar1976 I co-presented a demonstration of the tutorial algebra computer program I co-wrote at the NCTM Detroit meeting. April1975 I co-presented a talk and demonstration of the tutorial algebra computer program I co-wrote at the CBI Expo, Macomb Intermediate School District.

NSF Grants Sept 07 Aug 09 CPATH CB: Computing and Undergraduate Engineering: A Collaborative Process to Align Computing Education with Engineering Workforce Needs Jul 07 Jun13 EEES: Engaging Early Engineering Students to Expand Numbers of Degree Recipients Jan 10 Dec 12 CPACE II: Implementation of a Reformed Curriculum that Integrates Computational Thinking across Engineering Disciplines

Award Jan 1997 "Striving for Excellence" award from LCC and WLAJ-53ABC

Curriculum Work 2006 Development of CPSC131, "Numerical Methods and MATLAB"

Interest Integration of technology (graphing calculator and mathematical software) into the classroom to assist the students in understanding mathematical concepts and as a tool in problem solving.

# Computational Expertise in Engineering: Aligning Workforce Computing Needs with Computer Science Concepts


## Abstract

The 20th century ended with a multitude of engineering accomplishments that influenced and changed every aspect of human life. Globalization, international competition, an increasingly diverse population, and a rapid growth in computational capabilities and infrastructure are some of the challenges that will test the boundaries of engineering ingenuity in the 21st century. The Collaborative Process to Align Computing Education with Engineering Workforce Needs (CPACE) project team developed a collaborative *process* to identify the computational skills that are essential for a vital 21st century engineering workforce[1, 2]. Our objective is to revise the undergraduate engineering curricula to infuse computational problem-solving competencies—across engineering departmental courses. These competencies are aligned with industry needs and enable students to integrate conceptual knowledge, technical skills and professional practice. In this paper we describe the process that we used to translate our findings—computational competencies/needs in the engineering workplace—into fundamental computer science (**CS**) concepts that can be used in curricular implementation. We also discuss the initial phase of our curricular implementation strategy in two disciplinary engineering programs at Michigan State University (MSU) and transfer program at Lansing Community College (LCC).


## Project Implementation Strategy

Our project implementation strategy is based on the *transformation model* depicted in Figure 1, which comprises five interactive nodes:

- Node 1: Interview/survey engineering stakeholders to identify the computational competencies needed in the engineering workplace.
- Node 2: Abstract common—in an engineering context— computational problem-solving principles from the interview/survey data.
- Node 3: Align the computational problem-solving principles with computer science (CS) concepts.
- Node 4: Identify opportunities to integrate/reinforce these CS concepts in the curricula.
- Node 5: Implement revisions in engineering curricula.

The Transformation Model provides a framework that allows all stakeholders to see the interrelationships between what have, up to now, been discrete activities. The goal is to help each of the stakeholders view their needs in the context of this larger framework and to find ways to better engage all stakeholders in the entire process. This is a cyclic model with feedback among the five major nodes (dashed arrows). Given the rapid pace of technical change, the iterations and interactions through the nodes in the transformation model would continue, with increasingly better integration across all phases of the model.

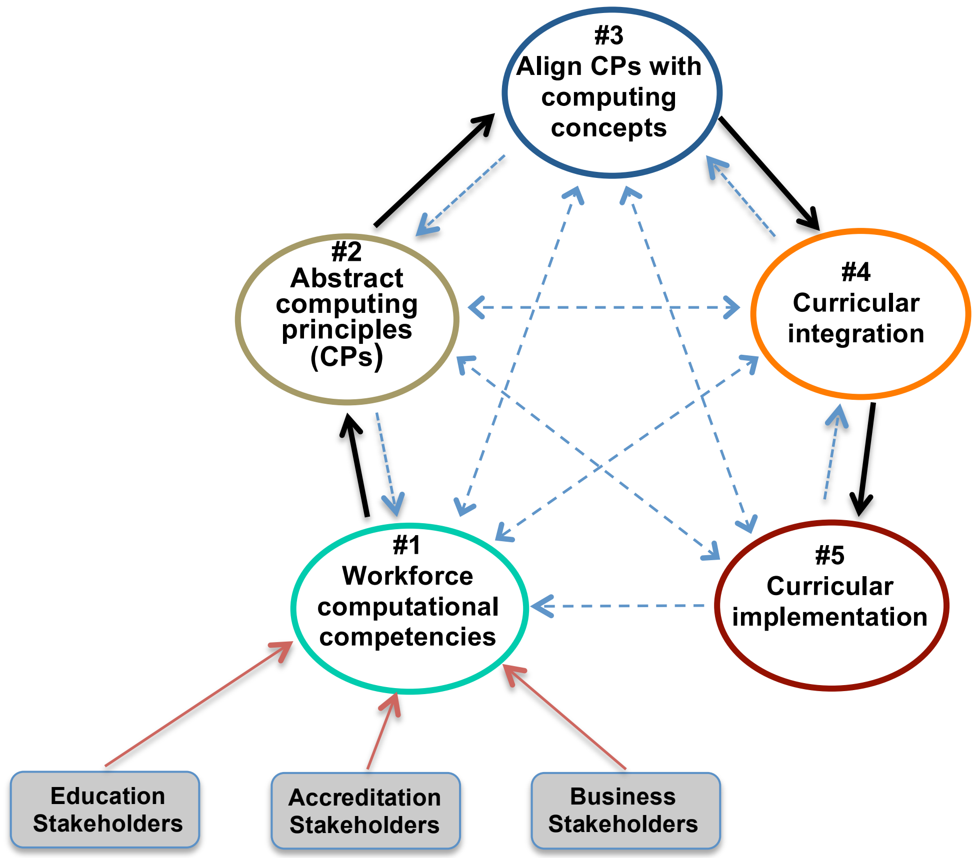


**Figure 1**. The Transformation Model provides a framework that allows all stakeholders to see the interrelationships among the different activities (nodes). The black solid arrows indicate the flow of [project] activities starting in node 1 through node 5. This is a cyclic model with feedback among the five major nodes as indicated by the blue dashed arrows.

**Workforce-Computing Needs**

As indicated in the transformation model (Figure1-node 1) we interviewed and surveyed engineering stakeholders to understand engineering workplace needs for computational competence both at the practical-tool level and at the computational thinking level. We interviewed the head of engineering, human resources executives (preferably both) to understand their employees' use of computer technology and the computational skills needed in their businesses; we conducted 27 interviews with companies representing a cross-section of engineering disciplines and different industry sectors [1, 2]. The main objectives of the employee surveys were 1) to understand what people working in engineering and technology feel are the strengths and weaknesses of their undergraduate computing education and 2) to identify current and future computational problem-solving gaps based on employee's views of future needs and trends. We conducted electronic surveys of 250 employees of participating companies [1, 2].

We organized the results of the interview and survey analyses in three general categories: general skills, computational skills and future of engineering practice. Table I presents a summary of our findings. In general employers: a) place a high value on interpersonal skills such as communication, ability to organize and present data, and the ability to function in a team; b) see critical and innovative thinking and problem solving as important attributes; c) see trends towards computational globalization, which translate to the need for engineers to understand business practices and the importance of integrating engineering data across larger systems.

**Table I**. Categories of skills identified by engineering stakeholders.

| General Skills | Computational Aspects | Future Engineering Practice |
|---|---|---|
| - Communication skills<br>- Team work<br>- Critical thinking<br>- Innovative thinking<br>- Problem solving (both conceptual and operational)<br>- Ability to learn/adapt | - Basic computational skills.<br>- Understanding of principles, application and limitations of computational tools<br>- Using technology to collaborate at all levels<br>- Use of technology to support broad problem solving and decision making<br>- Familiarity with multiple software systems<br>- Ability to move between abstractions in software and physical systems<br>- Multiple CAD programs including 3D modeling<br>- Process simulation packages<br>- Numeric computational platforms<br>- Excel (High level capabilities)<br>- MS Office<br>- Some programming | -Corporate development, leadership, management skills.<br>- Project management software<br>- Increasing integration of engineering data across larger systems<br>- More business intelligence embedded in systems<br>- Data Mining<br>- Globalization<br>- Environmental impact across disciplines. Design for the environment (DFE)<br>- Research and development including:<br>• Material development/new applications for existing material.<br>• Electronic communication.<br>• Next generation of technology<br>- Increasing use of simulation to reduce materials usage in design phase. |

Our results are consistent with other research on engineering education[3, 4] and details of the process and findings resulting from the completion of nodes 1 and 2 in the transformation model in figure 1 are presented elsewhere[1, 2]. In the sections below we describe the process that we used to align the engineering workforce-computing needs with CS concepts that can be used in curricular implementation (nodes 2-4). We also discuss how we are using this data-to-CS-concept alignment as a framework to design and implement curricular revisions.

## Workforce-Computing Needs *Alignment* to CS-Concepts

Based on employer interviews and employee surveys conducted in engineering businesses and industries we identified their needs for computational competencies. Since the computational competencies noted in our interview and survey data can be specialized to particular disciplines, industries, or even companies, we focused on identifying the underlying computational principles. These common principles incorporate key components of computational needs in the broad [workplace] engineering context. In other words we used these common principles to translate our interview and survey data into fundamental computer science (CS) concepts that can be integrated in the curricula. To accomplish this task we evaluated three different computational frameworks. Below we discuss some elements from each of the frameworks and discuss the issues that we encountered when trying to apply these frameworks to our interview and survey data.

1) In his *Great Principles of Computing*[5], Peter Denning adopts the terms 'Computing Mechanics' to group the structure and operations of computations. He refers to the principles of a field as "a set of interwoven stories about the structure and behavior of field elements." He groups the stories of the computing field into five categories [principles]: computation, communication, coordination, automation, and recollection; the lines between these categories overlap and any given element can fall within various categories. In his portrait of computing Denning incorporates computing mechanics, design principles and computing practices—one of which is engineering systems.

The depth of Denning's characterization, which includes not only the computing principles but also computing practices and core technologies, aligns seamlessly with curricula for CS majors. Indeed its focus on computing as a discipline made it difficult to apply to our interview and survey data, which reflects the use of computational tools and computational thinking in the context of the engineering workplace.

2) Jeannette Wing's discussion of *Computational Thinking* (CT)[6] can be summarized in terms of eight exemplar categories:

- Building on power and limits of computing processes.
- Solving problems, designing systems, and understanding human behavior.
- Reformulating a difficult problem into one we can solve.
- Thinking recursively.
- Using abstraction and decomposition.
- Thinking in terms of prevention, protection and recovery from worst-case scenarios.
- Using heuristic reasoning to discover a solution.
- Complementing and combining math and engineering thinking.

We aligned our interview and survey data to some of these exemplars—those CT activities that are relevant in an engineering context. Upon completion of the alignment, it was clear that Wing's CT exemplars were too general to move from the common principles—from our interview and survey data—into fundamental computer science (CS) concepts that can be used for curricular revisions. For example the problem-solving category is too general and several competencies derived from our interview and survey analyses fit within this category. Using the FITness principles—as explained below— we were able to assign these competencies into more fitting categories.

3) *Being Fluent with Information Technology Report* (FITness report)[7]. The concepts identified in the FITness report outline the basic ideas and principles underpinning CS. The fundamental nature of these concepts notwithstanding, they are instantiated in practical technologies and applications that allowed us to move from the computational competencies identified in our

industrial data to CS concepts that can be integrated in the curricula. The CS concepts enumerated in the Fluency with Information Technology (FITness) report offered the best framework to complete our alignment; Table II shows an example using the data that we collected from one of our companies and aligning it to the FITness principles.

Operationally, each member of the research team used this framework to categorize the data. This was followed by a group discussion to reach a consensus alignment. The process was iterative until all the data were analyzed. We used excel to create a matrix mapping the interview responses from each company [rows] to each of the FITness categories [columns] (Appendix 1). At the end of this mapping we counted the cells containing text for each interviewed company and under each FITness category. A complete alignment of all the data showing the text counts is included in Appendix 1.

**Table II**. Alignment of engineering computational competencies to FITness report concepts

| **Industrial Data from Interviews and Surveys** | |
|---|---|
| *Mission Critical Themes (reported by employers)* | *Computational Competencies* (reported by employers)* |
| Focus is launching new products, from concept to production. Sharing information, design, and computations. Develop ideas into parts. Mold flow analysis is very important. CAD design and being able to analyze the designs regardless of the software. (This highlights the fact that we work with different types of customers and need to accommodate all of them and come up with the product that the customer wants even if the customer has less definition about the product). | Mold flow analysis, simulations from CAD drawings, CAD design, Multiple CAD* programs, MS Office tools, word, outlook, power point, excel how to use/manipulate complex spreadsheets, FEA software, send CAD to tool-shops. Try to use particular packages but may need to use IGES to translate. Homegrown DB for monitoring manufacturing process. QID, Manufacturing Pro, Pilgrim QS software, MS Project, GANTT charts. |
| **Alignment to FITness Principles** | |
| *FITness Principles*** | *Alignment to Interview and Survey Data* |
| Information Systems | Homegrown DB for monitoring manufacturing process. |
| Digital Representation of Information | Try to use particular packages but may need to use IGES to translate. |
| Information Organization | MS Office tools, word, outlook, power point, excel how to use/manipulate complex spreadsheets. |
| Modeling and Abstraction | Mold flow analysis, simulations from CAD drawings, CAD design, Multiple CAD* programs. |
| Algorithmic Thinking and Programming | Excel how to use/manipulate complex spreadsheets. |

* The computational competencies are color coded to indicate how the alignment to the particular FITness principle was done.
** For a complete description of each FITness principle please refer to Appendix 2

The chart in Figure 2 is the result of the final alignment of all the interview data. The percentages are based on counts of numbers of cells in columns containing text (Appendix 1). The chart shows the distribution of the computational competencies—required in the engineering workplace—mapped to CS concepts that can be used to implement curricular changes in engineering courses.

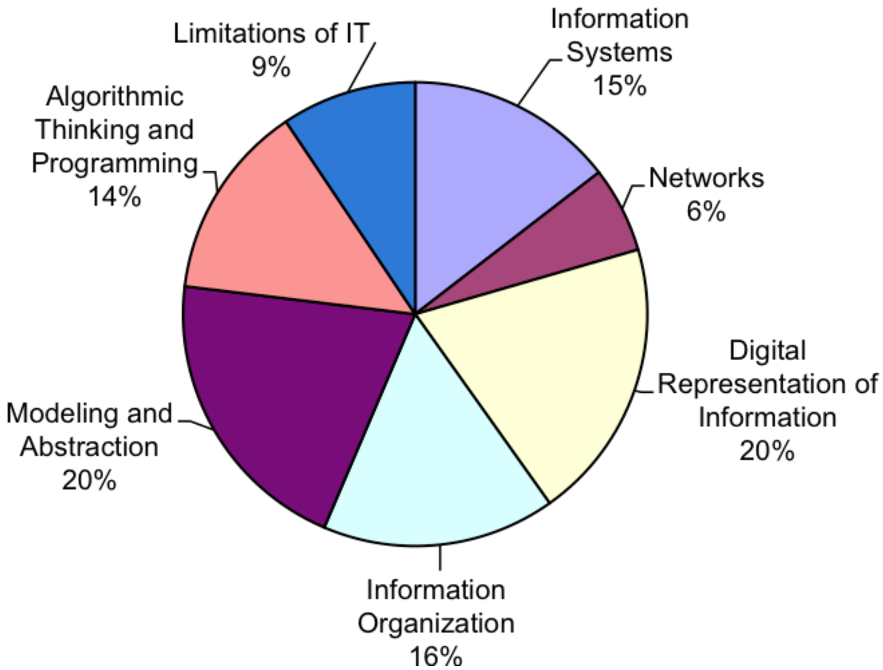


**Figure 2**. Distribution of engineering workplace computational competencies aligned to computer science concepts. The percentages are based on counts of numbers of cells in columns containing text (see the text for details).

**Curricular Implementation Strategy**

Our goal is to better align our engineering graduates capabilities—to solve disciplinary problems utilizing computational skills—with the needs of industrial stakeholders. To accomplish this, we are using this data-to-CS-concept distribution (Figure 2) as a framework to implement curricular revisions in two test-bed programs at MSU and LCC. We are mapping the concepts across all four years of the engineering curricula beginning with two engineering disciplines, Chemical and Civil, at MSU and pre-engineering courses (transfer) at LCC.

Our objective is to introduce a series of authentic engineering problems that provide a context where students are required to apply the various computational concepts for their solution. We are developing the problems in consultation with stakeholders from industry, and faculty from engineering disciplines to ensure that they exemplify relevant industrial scenarios within the discipline.

We are currently identifying problems that are appropriate to a variety of courses and can be used with varying degrees of complexity depending upon the course level. First-year courses would use simplified versions of problems. As students progress through their programs, the problems will become more complex. However, the underlying computing concepts— fitting course objectives—will be explicitly addressed across the various courses and throughout the degree program. A generic example of concept distribution and course mapping across the four years of the engineering curricula is depicted in table III.

**Table III**. CS Concept distribution across engineering curricula

| | CS Concept 1 | CS Concept 2 | CS Concept 3 | …CS Concept N |
|---|---|---|---|---|
| Transfer* | Target Course A | | Target Course B | |
| Freshman | Target Course C | Target Course D | | Target Course E |
| Sophomore | | Target Course F | | Target Course G |
| Junior | | Target Course H | Target Course I | |
| Senior | Course J | Course K | Course L | Course M |

*Refers to transfer students from <community college>

**Summary and Future Directions**

To prepare graduates to flourish in the global economy of the 21st century, engineering educators need to design curricula that incorporate innovation and flexibility based on constituency input and quality improvement principles [8]. The CPACE project team addresses these challenges in the context of computing education within engineering disciplines. CPACE brings together post secondary educators and business, industry and community leaders in a collaborative process to transform undergraduate computing education within the engineering and technology fields. We have created a partnership between engineering stakeholders from multiple sectors to identify the needs for computational problem-solving competencies in the engineering workplace, to define how these competencies can be integrated across curricula, and to revise the curricula to integrate these competencies across all four years of the engineering curricula[1, 2].

Based on the results of our employer interviews and employee surveys we developed an understanding of industry needs with regard to computational competencies both at the practical-tool level and at the computational problem-solving level. We aligned these data to computer science (CS) concepts that can be used to guide curriculum reforms (Figure 2). We are using this CS concept distribution to guide our design and implementation of the curricular reform. Our objective is to vertically integrate authentic problems that exemplify computational problem solving within the disciplines. The goal is for engineering graduates to enter the workforce with improved and practice-ready computational competencies that are aligned with industry needs and enables them to understand computational problem-solving in the context of the principles of computer science. The reform is beginning in two academic majors at MSU, chemical and civil engineering, and pre-engineering transfer courses at LCC. We expect to complete the implementation plan in the target courses in Fall 2012.

Our working hypothesis is that students going through the target courses sequences in civil engineering and in chemical engineering prior to implementation of our modules (control groups) will apply fewer examples of computational problem-solving competencies in the senior capstone course within their discipline (either civil engineering or chemical engineering) as compared to those students who take the target courses with our implemented modules (treatment groups). Our approach includes collecting quantitative and qualitative data for both treatment and comparison groups.

Our future directions include:

- Continue identifying authentic engineering problems.
- Complete the instructional design for the target courses in chemical and civil engineering at MSU and targeted courses at LCC.
- Develop appropriate instructional materials to support implementation by the disciplinary faculty who teach the target courses.
- Continue collecting quantitative and qualitative data.

In a broadest context, our project is an exploration in institutional change necessary for sustaining [our] curricular innovations after the funding ends. A central consideration of this project is the implementation of an effective change strategy that allows the successful adoption of the reform beyond classroom, individual faculty and ideally beyond institutions. This dimension of the project will be discussed in a forthcoming publication.

## Bibliography

[1] Vergara, C. E., Urban-Lurain, M., Dresen, C., Coxen, T., MacFarlane, T., Frazier, K., et al. (2009). Leveraging workforce needs to inform curricular change in computing education for engineering: The CPACE project. Computers in Education Journal, Vol XVIIII (4), 84-98.

[2] Vergara, C. E., Urban-Lurain, M., Dresen, C., Coxen, T., MacFarlane, T., Frazier, K., et al. (2009, October 18-21). Aligning computing education with engineering workforce computational needs: New curricular directions to improve computational thinking in engineering graduates. Paper presented at the Frontiers in Education, San Antonio, TX.

[3] Committee on the Engineer of 2020, Educating the engineer of 2020: Adapting engineering education to the new century. National Academy Press: Washington, DC, 2005

[4] Educating Engineers: Designing for the future of the field. The Carnegie Foundation for the Advancement of Teaching 2008.

[5] Peter Denning (Nov. 2003). Great Principles of Computing. Communications of the ACM, Vol. 46, (11), p. 15.

[6] Jeanette Wing. (Mar. 2006). Computational Thinking, Communications of the ACM, Vol. 49, (3), p 33.

[7] Being Fluent with Information Technology Committee on Information Technology Literacy, National Research Council. (1999).

[8] Lattuca, L. R., Terenzini, P. T., & Volkwein, J. F. (2006). Engineering change: A study of the impact of EC2000. Baltimore, MD: ABET, Inc.

**FITness Categories**

| Company | Information Systems | Networks | Digital Representation of Information | Information Organization | Modeling and Abstraction | Algorithmic Thinking & Programming | Limitations of IT |
|---|---|---|---|---|---|---|---|
| 1 | 51 | 0 | 68 | 119 | 87 | 48 | 0 |
| 2 | 53 | 0 | 3 | 67 | 108 | 100 | 0 |
| 3 | 144 | 0 | 110 | 42 | 347 | 0 | 100 |
| 4 | 0 | 26 | 162 | 93 | 162 | 0 | 0 |
| 5 | 98 | 0 | 7 | 100 | 167 | 74 | 45 |
| 6 | 36 | 36 | 95 | 192 | 266 | 76 | 112 |
| 7 | 287 | 287 | 287 | 287 | 8 | 31 | 111 |
| 8 | 210 | 0 | 121 | 125 | 238 | 132 | 0 |
| 9 | 292 | 0 | 34 | 43 | 106 | 58 | 0 |
| 10 | 365 | 142 | 11 | 88 | 276 | 48 | 0 |
| 11 | 186 | 29 | 41 | 205 | 271 | 16 | 0 |
| 12 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 13 | 0 | 0 | 24 | 133 | 201 | 0 | 0 |
| 14 | 0 | 0 | 3 | 0 | 309 | 75 | 0 |
| 15 | 50 | 0 | 7 | 0 | 102 | 0 | 0 |
| 16 | 222 | 108 | 137 | 61 | 103 | 195 | 243 |
| 17 | 30 | 0 | 130 | 126 | 156 | 0 | 294 |
| 18 | 0 | 0 | 0 | 0 | 32 | 58 | 164 |
| 19 | 505 | 0 | 11 | 0 | 57 | 0 | 0 |
| 20 | 0 | 0 | 23 | 46 | 228 | 60 | 0 |
| 21 | 0 | 0 | 31 | 0 | 461 | 6 | 180 |
| 22 | 0 | 0 | 9 | 90 | 203 | 38 | 0 |
| 23 | 43 | 0 | 27 | 215 | 242 | 0 | 188 |
| 24 | 53 | 0 | 38 | 96 | 279 | 153 | 67 |
| 25 | 130 | 61 | 34 | 126 | 503 | 0 | 330 |
| **COUNTA** | **17** | **7** | **23** | **19** | **24** | **16** | **11** |

## Appendix 2

Contains the list and definitions of the computational concepts used to create the Workforce-Computing Needs *Alignment* to CS-Concepts. The list of concepts is taken from the FITness report[7] (pg. 29).

***Computers***
Key aspects of a stored-program computer, including:

- The program as a sequence of steps,
- The process of program interpretation,
- The memory as a repository for program and data (including notions of memory hierarchy and associated ideas of permanence / volatility), and
- Overall organization, including relationship to peripheral devices (e.g., I / O devices).

The appropriate emphasis is not necessarily a specific electronic realization such as a particular computer, but rather the idea of a computational task as a discrete sequence of steps, the deterministic interpretation of instructions, instruction sequencing and control flow, and the distinction between name and value. Computers do what the program tells them to do given particular input, and if a computer exhibits a particular capability, it is because someone figured out how to break the task into a sequence of basic steps, i.e., how to program it.

***Information systems***
The general structural features of an information system, including, among others, the hardware and software components, people and processes, interfaces (both technology interfaces and human-computer interfaces), databases, transactions, consistency, availability, persistent storage, archiving, audit trails, security and privacy and their technological underpinnings.
Most knowledge workers in the labor force interact with one or more information systems, becoming knowledgeable about their characteristics and idiosyncrasies. Understanding the abstract structure of such systems prepares students for employment, enhances job mobility, enables workers to adapt to new systems more quickly, and helps them to exploit more fully the facilities of a given system.

***Networks***
Key attributes and aspects of information networks, including their physical structure (messages, packets, switching, routing, addressing, congestion, local area networks (LANs), wide area networks (WANs), bandwidth, latency, point-to-point communication, multicast, broadcast, Ethernet, mobility), and logical structure (client / server, interfaces, layered protocols, standards, network services).
Computers are generally much more useful when connected to each other and to the Internet. The goal is to understand how computers can be connected to each other and to networks, and how information is routed between computers. The appropriate emphasis is how the parameters of communication, such as latency and bandwidth, affect the responsiveness of a network from a user's point of view and how they might limit one's ability to work.

***Digital representation of information***
The general concept of information encoding in binary form. Different information encodings: ASCII, digital sound, images, and video / movies. Topics such as precision, conversion and

interoperability (e.g., of file formats), resolution, fidelity, transformation, compression, and encryption are related, as is standardization of representations to support communication.
The appropriate emphasis is the notion that information that is processed by computers and communication systems is represented by bits (i.e., binary digits). Such a representation is a uniform way for computers and communication systems to store and transmit all information; information can be synthesized without a master analog source simply by creating the bits and so can be used to produce everything from *Toy Story* animations to forged e-mail; symbolic information in machine-readable form is more easily searchable than physical information.

***Information organization***
The general concepts of information organization, including forms, structure, classification and indexing, searching and retrieving, assessing information quality, authoring and presentation, and citation. Search engines for text, images, video, audio.
Information in computers, databases, libraries, and elsewhere must be structured to be accessible and useful. How the data should be organized and indexed depends critically on how users will describe the information sought (and vice versa), and how completely that description can be specified. In addition to locating and structuring information, it is important to be able to judge the quality (accuracy, authoritativeness, and so forth) of information both stored and retrieved. Section 3.2 provides some additional discussion.

***Modeling and abstraction***
The general methods and techniques for representing real-world phenomena as computer models, first in appropriate forms such as systems of equations, graphs, and relationships, and then in appropriate programming objects such as arrays or lists or procedures. Topics include continuous and discrete models, discrete time, events, randomization, and convergence, as well as the use of abstraction to hide irrelevant detail.
Computers can be made to play chess, predict the weather, and simulate the crash of a sports car by abstracting real-world phenomena and manipulating those abstractions using transformations that duplicate or approximate the real-world processes. One goal is understanding the relationship between reality and its representation, including notions of approximation, validity, and limitations; i.e., not all aspects of the real world are modeled in any one program, and a model is not reality.

***Algorithmic thinking and programming***
The general concepts of algorithmic thinking, including functional decomposition, repetition (iteration and / or recursion), basic data organizations (record, array, list), generalization and parameterization, algorithm vs. program, top-down design, and refinement. Note also that some types of algorithmic thinking do not necessarily require the use or understanding of sophisticated mathematics. The role of programming, which is a specific instantiation of algorithmic thinking, is discussed in Chapter 3.
Algorithmic thinking is key to understanding many aspects of information technology. Specifically, it is essential to comprehending how and why information technology systems work as they do. To troubleshoot or debug a problem in an information technology system, application, or operation, it is essential to have some expectation of what the proper behavior should be, and how it might fail to be realized. Further, algorithmic thinking is key to applying information technology to other personally relevant situations.

***Universality***
The "universality of computers" is one of the fundamental facts of information technology discovered by computing pioneers A.M. Turing and Alonzo Church in the 1930s, before practical computers were created. Shorn of its theoretical formalism and expressed informally, universality says that any computational task can be performed by any computer. The statement has several implications:

- No computational task is so complex that it cannot be decomposed into instructions suitable for the most basic computer.
- The instruction repertoire of a computer is largely unimportant in terms of giving it power since any missing instruction types can be programmed using the instructions the machine does have.
- Computers differ by how quickly they solve a problem, not whether they can solve the problem.
- Programs, which direct the instruction-following components of a computer to realize a computation, are the key.

***Limitations of information technology***
The general notions of complexity, growth rates, scale, tractability, decidability, and state explosion combine to express some of the limitations of information technology. Tangible connections should be made to applications, such as text search, sorting, scheduling, and debugging.

Computers possess no intuition, creativity, imagination, or magic. Though extraordinary in their scope and application, information technology systems cannot do everything. Some tasks, such as calculating the closing price for a given stock on the NASDAQ exchange, are not solvable by computer. Other tasks, such as that of placing objects into a container so as to maximize the number that can be stored within it (e.g., optimally filling boxcars, shipping containers, moving vans, or space shuttles), can be solved only for small problems but not for large ones or those of practical importance. Some tasks are so easily solved that it hardly matters which solution is used. And, because the programs that run on computers are designed by human beings, they reflect the assumptions that their designers build into them, assumptions that may be inappropriate or wrong. Thus, for example, a computer simulation of some "real" phenomenon may or may not accurately reflect the underlying reality (and a naïve user may be unable to tell the difference between a generally true simulation and one that is fundamentally misleading). Assessing what information technology can be applied—and when it should be applied—is essential in today's information age.