Towards Improving Computational Competencies for Undergraduate Engineering Students

Dr. Claudia Elena Vergara, Michigan State University

Claudia Elena Vergara is a Research Scientist in The Center for Engineering Education Research (CEER). She received her Ph.D. in Plant Biology from Purdue University. Her scholarly interests include: improvement of STEM teaching and learning processes in higher education, and institutional change strategies to address the problems and solutions of educational reforms considering the situational context of the participants involved in the reforms. She is involved in several research projects focusing on competencies-based curriculum redesign and implementation aimed to integration across curricula; increasing the retention rate of early engineering students; providing opportunities for STEM graduate students to have mentored teaching experiences.

Dr. Mark Urban-Lurain, Michigan State University

Mark Urban-Lurain is an Associate Professor and Associate Director of the Center for Engineering Education Research at Michigan State University.

Dr. Urban-Lurain is responsible for teaching, research and curriculum development, with emphasis on engineering education and, more broadly, STEM education.

His research interests are in theories of cognition, how these theories inform the design of instruction, how we might best design instructional technology within those frameworks, and how the research and development of instructional technologies can inform our theories of cognition. He is also interested in preparing future STEM faculty for teaching, incorporating instructional technology as part of instructional design, and STEM education improvement and reform.

Dr. Jon Sticklen, Michigan State University

Jon Sticklen is the Director of the Center for Engineering Education Research at Michigan State University. He also serves MSU as Director of Applied Engineering Sciences, an undergraduate bachelor of science degree program that is highly interdisciplinary focusing on both engineering and business. He also is a faculty member in the Department of Computer Science and Engineering. In the decade of the 90s, Dr. Sticklen founded and led a computer science laboratory in knowledge-based systems focused on task specific approaches to problem solving, better known as expert systems. Over the last decade, Dr. Sticklen has pursued engineering education research focused on early engineering with an emphasis on hybrid course design and problem-based learning; his current research is supported by NSF/DUE and NSF/CISE.

Prof. Abdol-Hossein Esfahanian, Michigan State University
Hannah McQuade, The Center for Engineering Education Research
Andrew League, Michigan State University
Mr. Chris John Bush, Center for Engineering Education Research
Mr. Michael Cavanaugh, Michigan State University: Center for Engineering Education Research

Michael Cavanaugh is a third year graduate student in the Anthropology Department at Michigan State University. He has a B.A. in American Indian Studies from the University of Wisconsin Eau-Claire. He earned an M.A. in American Indian Studies from UCLA. His Ph.D. research at MSU is focused on American Indian experiences in postsecondary education and the retention of Native students in STEM disciplines. He currently works for the Center for Engineering Education Research (CEER) at MSU. While at CEER he has worked on the CPACE Project (Collaborative Process to Align Computing Education with Engineering Workforce Needs) and helped to build a framework for assessing computational competencies within engineering education.

Enhancement of Computational Competencies Across Engineering Curricula

Abstract

In the global economy of the 21st century, the preparation of a competitive U.S. workforce with knowledge and understanding of critical computing concepts, methodologies, and techniques is essential. The Collaborative Process to Align Computing Education with Engineering Workforce Needs (CPACE) team developed a partnership among various stakeholders – Michigan State University (MSU) and Lansing Community College (LCC) and business and industry leaders – to identify the computational skills that are essential for a globally competitive engineering workforce. Our goal is to redesign the role of computing within the engineering programs at MSU and LCC to develop computational competencies –informed by industry needs– by infusing computational learning opportunities into the undergraduate engineering curriculum. The emphasis of this paper is on the curricular implementation phase (CPACE II) focusing on:

- Preliminary survey results across target courses at MSU.
- Efforts to assess students' computational competencies using a rubric based on the CPACE computational competencies.
- Brief discussion of our efforts to develop and validate assessments to measure computational competencies for engineering students.

Introduction

The learning sciences have influenced repeated calls for improving engineering education that focus on providing students with the opportunities to integrate their knowledge across disciplines through authentic problem solving [1-6]. Computation for engineering cannot simply be addressed with one or two courses in computing or a few examples scattered in the curriculum, but must be integrated as part of an engineer's training to become a "Holistic Engineer" [7].

One of the challenges of preparing engineers for the rapidly changing workplace is to provide the foundations they need to choose among and use a wide range of changing computational tools. To address the computational needs of engineers, we must ground the curricula in problems from engineering while helping students acquire the competence to understand underlying principles and abstractions that will help them solve new problems using new tools. Our educational challenge is to help engineering students move from *novice* understanding of computing that focuses on the superficial operation of particular software, towards *competence* expertise that is grounded in a deeper conceptual understanding relevant to their engineering discipline [8]. Drawing upon the extensive literature on how people learn [9], particularly in engineering [10-12], our goal is to infuse computational competencies throughout the engineering curricula by integrating problems of disciplinary engineering practice.

CPACE Project Overview

The CPACE project is divided in two phases, CPACE I and II. During CPACE I we: a) identified the computational competencies needed in the engineering workplace; b) developed a 'data-to-computer science (CS)-concept map' to translate our research findings into fundamental CS concepts that can be used in curricular implementation. Our results are consistent with other research on engineering education[13, 14] and details of the process and findings from CPACE I are presented elsewhere[15, 16].

CPACE I: Workforce-Computing Needs

To understand engineering workplace needs for computational competence both at the practical-tool level and at the computational problem solving level, we interviewed and surveyed engineering stakeholders. We organized the results of the interview and survey analyses in three categories: general skills, computational skills and future of engineering practice. Employers place high value on: a) interpersonal skills such as communication, ability to organize and present data, and the ability to function in a team; b) critical and innovative thinking and problem solving; and (c) employers see trends towards computational globalization which translates to the need for engineers to understand business practices and the importance of integrating engineering data across larger systems. The ability of engineers to understand computational principles in the context of the engineering practice allows them to select and use computation to solve engineering problems. With regard to specific software, Excel, design and modeling software, and data and project management software were identified as very important to the engineering practice[13, 14]. Table 1 presents a summary of the skills identified by engineering stakeholders.

Table 1. Categories of skills identified by engineering stakeholders

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

CPACE I: Workforce-Computing Needs Alignment to CS-Concepts

To translate our employer data into computer science (CS) concepts that can be integrated in the curricula, we focused on identifying the underlying computational principles. These computational principles incorporate key components of computational needs in the broad [workplace] engineering context. Details of the process and findings from this phase of the project are presented elsewhere[17]. The chart in Figure 1 shows the distribution of the computational competencies – required in the engineering workplace – mapped to CS concepts that can be used to implement curricular changes. Our goal is to better align our engineering graduates computational problem solving capabilities with the needs of industrial stakeholders represented in this distribution.

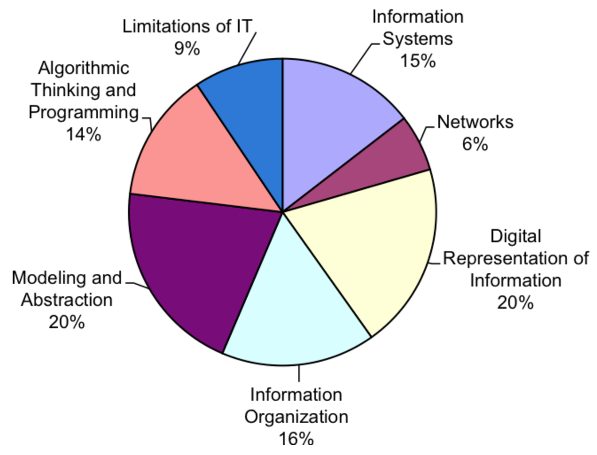

Figure 1. Distribution of engineering workplace computational competencies aligned to CS concepts[17].

The second and current phase of the project (CPACE II) comprises the implementation of curricular revisions in two engineering disciplines Chemical and Civil at MSU and pre-engineering courses at LCC. In this paper we focus on a subset of the survey analyses and discuss our efforts to analyze student artifacts using a rubric that we created based on the CPACE computational competencies depicted in Figure 1. We will briefly discuss our efforts to develop and validate assessments to measure computational competencies for engineering students.

Curricular Implementation (CPACE II)

In CPACE II our goal is to infuse *computational problem-solving* competencies throughout the curricula[18, 19]. To achieve this goal and bring about an integrated computing experience, our strategy entails using problems derived from contemporary industrial engineering practice. The problems are developed in consultation with stakeholders from industry, and faculty from engineering and CS to ensure that they exemplify relevant industrial scenarios within the discipline. Together the team will form the educational experience to further the development of computational problem solving in the students. These problems provide a context where students are required to apply various computational concepts for their solution. Initial implementation includes Chemical (CHE) and Civil (CE) Engineering at MSU and pre-engineering courses at LCC. We targeted courses across all four years of the engineering curricula. Table 2 summarizes the courses targeted at MSU.

Table 2. Description of courses targeted at Michigan State University

Course name	**Level**	**Description**	**Approximate annual enrolment**
Engineering modeling (EGR 102)	First year	Intense computational course part of <university's> First Year Engineering Program.	900
Civil Eng. (CE) Statics (CE 221)	Sophomore	Vector description of forces and moments. 2 and 3- D equilibrium of particles and rigid bodies. Analysis of trusses, frames, and machines. Coulomb friction.	700
CE Str. Anal & Design (CE 305)	Junior	Theory of structural analysis for statically determinate structures. Qualitative structural analysis and behavior. Load estimation and placement. Introduction to structural analysis computer software. Introduction to statically indeterminate structures.	120
CE Intro Fluid Mech. (CE 321)	Junior	Fluid properties, fluid statics, fluids in motion. Conservation of mass, energy, and momentum. Dimensional analysis and similitude. Internal and external flows. Applications.	120
Chemical Eng. (CHE) Modeling and Analysis of Transport Phenomena (CHE 210)	Sophomore	Steady and unsteady state material and energy balances. Fluxes and rate processes. Shell balances. Balance equations for mass, heat, and momentum transport. Analogies among mass, heat, and momentum transport. Analytical and numerical solutions. Application of computational methods to problem solutions.	110
Mass Transfer and Separations (CHE 312)	Junior	Diffusion. Mass transfer coefficients. Design of countercurrent separation systems, both stage wise and continuous. Distillation, absorption, extraction. Multicomponent separations. Batch processes. Computer-aided design methods.	110
Thermodynamics for Chemical Engineering (CHE 321)	Junior	First and second laws. Thermodynamics of flow and energy conversion processes. Properties of single and multi-component systems. Phase equilibria. Chemical equilibria in reacting systems.	110

Data Collection and Analyses

The Introduction to Engineering Modeling (EGR 102) is one of the cornerstone academic components of the first year engineering experience at MSU. Approximately 900 students, primarily freshmen, enroll in EGR 102 per academic year. The course focuses on the solution of engineering problems using computational tools (e.g., MATLAB and EXCEL). Because the course serves seven engineering disciplines, course instruction is focused on helping students develop *transferable computational expertise* that students can subsequently apply to the use of discipline-specific software in later coursework and their professional practice. Beginning with EGR 102 and the computational skills taught in EGR 102 we are interested in understanding how students' computational problem solving skills using tools such as MATLAB and excel progress through the curricula? Are students able to transfer their conceptual understanding and computational skills to solving problems that require the use of new tools and computational skills?

Following a mixed methods approach, quantitative and qualitative data are collected. We collected student surveys at the beginning and end of target courses; we also conducted student focus groups and interviews. Standard class data on learning outcomes and sample course work e.g. final report on the assigned problem and relevant homework assignments from the target courses at LCC and MSU were also collected and assessed using a computational competencies rubric that we designed.

To gauge how students' computational skills progress through the curricula, we created self-reported survey instruments that ask students about their confidence to perform specific tasks in each of the software packages. The surveys are constructed and administered online. The instrument included open- and close-ended items. The goals of the student surveys were to: (a) measure general attitudes towards engineering; (b) measure attitudes towards computational problem solving; and (c) determine the use and application of computational tools. The surveys for all the target courses included the same MATLAB and excel related questions. Questions about specific software tools required in a particular course were also included in consultation with the instructor (e.g. SAP 2000 for CE 305 and ASPEN for CHE 312).

Test/retest of these instruments were performed with a population of teachers and pre-service teachers as part of a project aimed to improve the integration of technology in a teacher education program by promoting Fluency with Information Technology (FITness)[20]. We performed factor analysis (FA) to determine how many factors the survey questions load on. The FA on items for Excel together or MATLAB together revealed that they load only on single factors (one excel, one MATLAB). For all survey analyses we used SPSS V20 and performed 1000 bootstrap iterations to estimate confidence intervals for all regression coefficients, using a 95% confidence interval for significance. The open-ended responses were analyzed using the SPSS Text Analysis for Surveys.

To complement our analyses and understand the complexity of the students' computational thinking processes we conducted semi-structured interviews and focus groups; the objective of the interviews was to focus on the role of computation during the problem solving process. During the interviews students were asked to describe/recount their thought processes as they solved the engineering problem. The interviews were recorded and transcribed verbatim; the transcripts were segmented and coded using Atlas Ti software tool for the analyses of qualitative data. We are currently completing the analyses of the interview data; complete and detailed results of this qualitative data set will be presented in a forthcoming manuscript.

The majority of the data presented in this paper corresponds to survey data collected from students beginning in Fall, 2011, through Spring, 2013. There are brief references to preliminary interview data as we are still completing those analyses.

Survey Analyses

These analyses include survey data collected from students beginning in Fall, 2011, through Spring, 2013. The courses included in these analyses are EGR102 (freshman level), CHE 210 (sophomore level) and CHE 312 (junior level). As indicated above student surveys are administered at the beginning (pre) and end (post) of EGR 102, CHE 210, and CHE 312. This allowed us to track the *same students* as they moved from EGR 102 through the subsequent courses.

At MSU the one formal place in the engineering curriculum where students are explicitly taught computational tools is EGR 102; in this course students learn to use Excel and MATLAB to compute various numeric methods. While students in EGR 102 are graded in part on their ability to use these software packages, grades in their subsequent engineering courses do not explicitly evaluate their skill using these tools. To gauge how students' skills with these tools progress through the curricula, we created self-reported survey instruments that ask students about their confidence to perform specific tasks in each of the software packages. Some example questions for Excel include "I can compute the average of a column of numbers using a built-in function" and "I can create a macro to perform a mathematical function for which I know the formula." For each question students are asked to rate their ability on a 1-5 scale where 1 is "I don't know what that means" to 5, "I know what it means and am certain I can do it".

We performed exploratory factor analysis on these instruments and determined that there is a single dimension on which all items load, this allowed us to create composite total scores for Excel and MATLAB by averaging their responses to all items on the instrument to account for students who may not answer every item.

Students in EGR 102 reported statistically significant gains in most computational skills related to MATLAB ($p < .005$ Wilcoxon signed rank test for related samples pre and post survey) and Excel ($p < .005$ Wilcoxon signed rank test for related samples pre and post survey). Some of these skills include writing FOR loops and nested FOR loops, and basic programming in MATLAB. These results indicate that participation in the course project is helping students feel more confident with respect to their computational abilities. The survey results for students at the end of the EGR 102 course correlated with the final course grade (Excel, $r=.11$, $p < .003$; Matlab, $r=.272$, $p < .0001$). Note that most of the grade in EGR 102 was based upon MATLAB, rather than Excel, assignments.

To determine how student abilities with Excel and MATLAB progress throughout the curricula, we administered these surveys to students at the beginning and end of EGR 102, CHE 210, and CHE 312 (freshman to junior). This allowed us to *track the same students* as they moved from EGR 102 through the subsequent disciplinary courses. EGR 102 is required for most students in the College of Engineering, but the Chemical Engineering courses are required only for Chemical Engineering students. Table 3 shows the numbers of students who completed the EGR 102 *AND* the CHE 210 surveys.

Table 3. Cross tabulation of students who completed both EGR 102 *AND* CHE 210 surveys

		EGR102_Crse_Term_Code					Total
		FS11	FS12	SS11	SS12	SS13	
CHE210_Crse_Term_Code		153	212	198	387	443	1393
	FS11	0	0	7	0	0	**7**
	FS12	**4**	0	5	8	0	**17**
	SS12	**9**	0	24	1	0	**34**
Total		**166**	212	234	396	443	1451

The columns show the numbers of students who took the EGR 102 survey each semester and the rows show the numbers of students who took the CHE 210 survey. Thus, **166** students took EGR 102 in Fall 2011. Of those students, **4** students took CHE210 in Fall 2012 and **9** took CHE 210 in Spring, 2012. Of the 1451 EGR 102 students, **58** students took CHE 210 (numbers highlighted in red in table 3).

Table 4 shows the same information for students who took EGR 102 and CHE 312. Note that a total of **49** students (numbers highlighted in red in table 4) took both EGR 102 and CHE 312 in this time period.

Table 4. Cross tabulation of students who completed both EGR 102 *AND* CHE 312 surveys

		EGR102_Crse_Term_Code					Total
		FS11	FS12	SS11	SS12	SS13	
		154	212	205	388	443	1402
CHE312_Crse_Term_Code	SS12	0	0	7	1	0	**8**
	SS13	12	0	22	7	0	**41**
Total		166	212	234	396	443	1451

We compared the results across courses moving from freshman (EGR 102) to sophomore (CHE210) and junior (CHE312) levels; figure 2 shows the mean scores for questions related to the use of excel for students who took the pre- and post-course surveys in EGR 102, CHE 210 and CHE 312. The error bars are the 95% confidence interval (CI). Note that the student confidence increases from the beginning of EGR 102 (EGR102 pre-excel total in figure 2) to the end of the course (EGR102 post-excel total in figure 2), but then reverts to pre-EGR 102 levels when they start CHE 210 (CHE210 pre-excel total in figure 2). Their confidence continues to increase across CHE 210 (usually taken in the sophomore year) and CHE 312 (usually taken in the junior year).

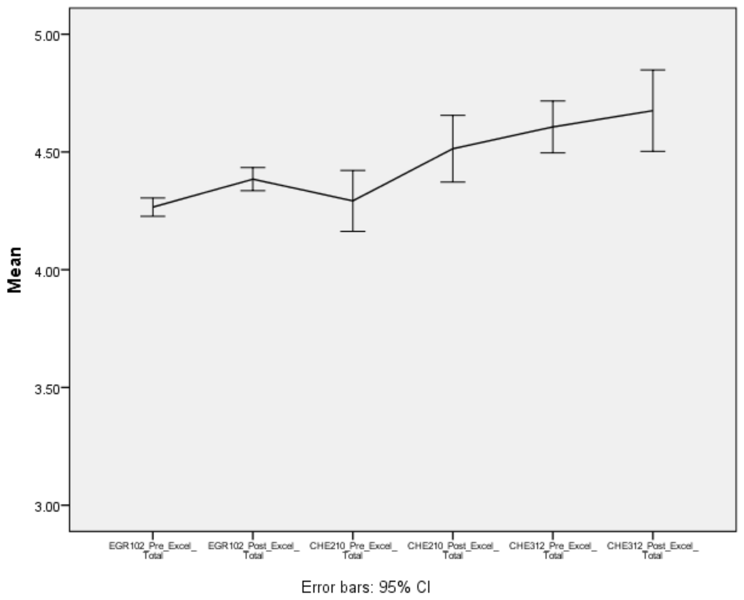

Figure 2. Mean excel scores for students who took the pre- and post-course surveys in EGR 102, CHE 210 and CHE 312.

Figure 3 shows the mean scores for questions related to the use of MATLAB. In this case, confidence in the use of MATLAB increases dramatically in EGR 102 (EGR102 pre and post MATLAB total in figure 3), but then decreases at the start of CHE 210. It increases by the end of CHE 210, but declines again at the start of CHE 312 and again increases. It is important to note that both CHE 210 and 312 courses provide students the opportunity to engage in course related activities such as projects and homework where they have the opportunity to practice their MATLAB skills.

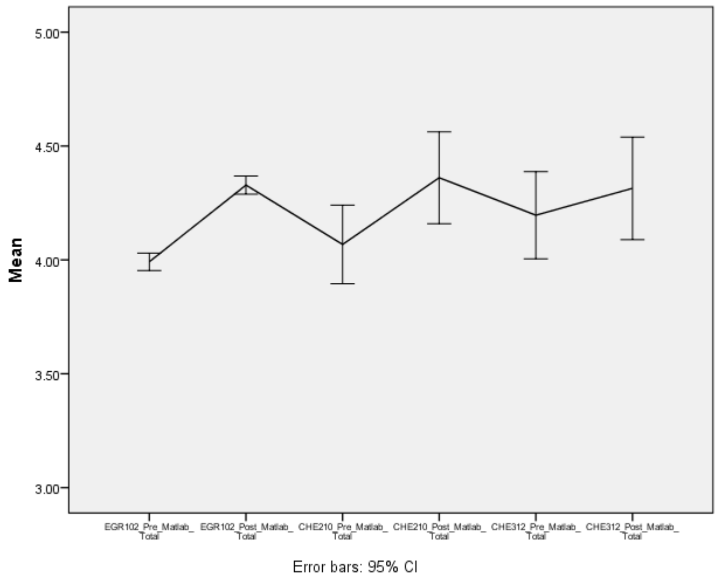

Figure 3. Mean MATLAB scores for students who took the pre- and post-course surveys in EGR 102, CHE 210 and CHE 312

There is a difference between the patterns observed for the MATLAB-related skills (figure 3) and the excel-related skills where students' self-reported confidence continues to increase across CHE 210 and CHE 312 (figure 2). A likely explanation comes from our interview data. Preliminary results from students' interviews indicate that they use Excel much more frequently in all of their disciplinary course work than they use MATLAB. They tend to use MATLAB only if the assignment specifically instructs them to, in contrast they will use Excel as their first choice when given the freedom to pick a tool.

Our preliminary survey analyses indicate that students who are engaged in an intense computational course such as the EGR 102 at MSU or the Numerical Methods and MATLAB course at LCC (results not shown) report statistically significant gains in the use and application of important computational skills and competencies after completing the course project. In all the examples analyzed students entering their disciplinary courses either at the sophomore or junior level report a drop in confidence in their MATLAB-related computational skills. Students in the disciplinary courses who are engaged in a project or classroom activities that encourage the use of those computational skills report regaining confidence in the use of those skills after completing the project. This is exemplified in both the sophomore and junior courses in CHE where students have the opportunity to practice their MATLAB skills in the context of the project and other course related activities.

CPACE Computational Rubric

To compare students' self-reported survey data regarding their computational abilities and their actual performance in classroom assignments, we designed and are currently developing rubrics based on the computational competencies aligned to CS concepts presented in figure 1. Table 5 shows two of the CPACE competencies from the distribution depicted in Figure 1 (Modeling & Abstraction and Algorithmic Thinking & Programming) and gives examples of identifying features (descriptors) aligned to associated activities and characteristics for each descriptor.

Table 5. Descriptors and Associated Activities/Characteristics for the Modeling & Abstraction and Algorithmic Thinking & Programming Competencies from Figure 1

Modeling & Abstraction	
Descriptors	**Activities/Characteristics**
Representation of a problem as a model (equation, graph, relationship, simulation model)	Use of appropriate assumptions and justifications. Identification of most crucial assumptions Parsimony. Use of correct approximations. Correct choice of variables
Implementation of the model (computational/mathematical approach) to obtain solution	Use of appropriate approach and applies generalizable solution. Reusability Correctness of the approach
Abstraction of computation	Appropriate decomposition of the implementation into small cohesive parts each of which is tested and debugged until complete Assembles and tests components for complete solution
Analyses of results/solution	Result is correct and recognized as such; self-corrected if necessary Meets specifications, codes or constraints
Algorithmic Thinking & Programming	

Descriptors	Activities/Characteristics
Use top-down design, and refinement to develop algorithms	Appropriate documentation of the design and use of high level descriptions of the solution before writing code (e.g use of flowcharts)
Selection of computational tools (e.g., programming language, software functions or features)	Selection of the most appropriate computational tool to implement the best solution
Use of data structures (record, array, list)	Matches characteristics of the data to the appropriate data structures (e.g., data types; structures, arrays) for efficient processing
Creation of generalizable solutions by parameterizing solutions	Writing of reusable code that utilizes parameters to specify individual cases
Good programming style	Writing of readable code; appropriate documentation, control: repetition (iteration and /or recursion), decision constructs
Program testing	Successful identification and debugging of problems

Initial pilot analyses using the rubric to assess and analyze student artifacts (homework assignment and course projects) revealed that employing the artifacts alone made it difficult to recognize students' computational thinking processes [let alone measure computational competencies] as they solved engineering problems.

Interview Analyses

One of the most challenging aspects during the implementation phase has been the lack of assessment instruments to measure student computational competencies. Initial analyses of student artifacts using the CPACE computational competencies rubric revealed that to understand the students' computational thinking processes as they solve engineering problems, we needed to complement the artifact analyses with student interviews.

We conducted one-hour semi-structured interviews with students enrolled in the target courses. Our objectives were to better understand the process by which students solve engineering problems using computational tools, specifically how, when and why computational tools are employed during the problem solving process. Students described their thought process as they solved the engineering problem using their course projects as a guide for both interviewer and interviewee.

Interview analyses have enabled us to more clearly align the CPACE computational competencies [using the rubric] and the student artifacts. Hence we have a better understanding about the ways in which students use computational skills as they solve engineering problems (table 6). While these are preliminary analyses this strategy is helping us triangulate between the different data sources i.e. self-reported students' surveys, students' artifacts, computational rubric and interviews. In particular the rich interview data have enabled us to better align the CPACE computational competencies [using the rubric] and the student artifacts including projects and other course assignments.

Our preliminary interview analyses agree with the results observed in the surveys; in general, students have limited conceptual understanding about underlying computational principles and fail to integrate their knowledge in ways that would allow them to extend what they learn in one context to new and different contexts.

Table 6. Sample interview data aligned to exemplar CPACE computational competencies (depicted in figure 1)

Exemplar Computational Competencies	Interview Quotes
Algorithmic thinking and programming: students who describe the use/function of particular computational tools and as a result demonstrate an understanding (or lack of) of the tool. How/why a particular tool works the way it does; references to programming, coding, debugging, design, iteration, parameterization, refinement, and utility of particular applications*.	"It's such a multicomponent system that to try to do it by hand would be just tedious … went to Polymath …hit go and it'll, it spits out a table and a graph if you want it." "MATLAB has more built-in functions in terms of math. Also, you know, Python is a powerful language, but it will require me as a programmer to code most of the functions and math myself, and that just leaves more room for error"
Digital representation of information: examples include references to: conversion, copy and pasting, migrating info, and representing data across fields*.	"[…] there's just so many things you need to input, to get Aspen to actually work, you know, the number of stages, the diameter of the column, the feed stage, the reflux ratio.."
Limitations of technology: examples include references to tool capabilities/purposes in relation to the problem; awareness of the assumptions made when simulating a 'real world' phenomenon; assessing what/when tool can be applied*.	"So instead of creating an Aspen document first and like getting our whole process flow in there and then trying to get it all to work, we figured it's probably just a better idea to, to tackle each individual process first and then try to like interconnect them in Aspen."
Modeling and abstraction: Examples include references to methods for representing 'real world' phenomena through computer modeling. References to representing a problem as a model (system equations, graphs, simulations) and implementing the model to obtain solution or solve problems (debug)*.	"From my basic knowledge, just being able to type in line then you're able to create a line which has a simple command. [...] you make a rectangle so those simple commands kind of add –gave me a sense of what I wanted to try."

* Descriptor from the rubric used to inform the interview coding process. The underlined items correspond to the computational competencies depicted in Figure 1.

Summary and Future Directions

The Collaborative Process to Align Computing Education with Engineering Workforce Needs (CPACE) project brings together MSU, LCC and business and industry leaders in an effort to transform undergraduate computing education within the engineering and technology fields. Based on the results of our employer interviews and employee surveys we developed a Computer Science concept distribution framework to guide the design and implementation of a curricular reform (Figure 1), beginning in two academic majors at MSU, chemical and civil engineering, and pre-engineering transfer courses at LCC. Our goal is to better align our engineering graduates capabilities – to solve disciplinary problems utilizing computational skills – with the needs of industrial stakeholders represented in this distribution. To accomplish an integrated computing experience, we integrate problems derived from contemporary industrial engineering into the target courses. These problems provide a context where students are required to apply various computational concepts for their solution. The underlying computing concepts – fitting course objectives – will be explicitly addressed across the various courses and throughout the degree programs.

Our analyses indicate that students who are engaged in project or classroom activities that encourage the use of computational skills report statistically significant gains in the use and application of important computational competencies. Our results also suggest that participation in the course project is helping students feel more confident with respect to their computational abilities. Very importantly both Chemical and Civil engineering students entering their disciplinary courses either at the sophomore or junior level reported a marked drop in confidence in their ability to use and apply important computational skills (e.g. MATLAB and EXCEL-related), and in all cases, student confidence declines the longer since they took EGR 102. A major conclusion of our study is that students have difficulties making connections between different computational tools and skills and fail to integrate their knowledge in ways that would allow them to extend what they learn in one context to new and different contexts. This points to a limited understanding about underlying computational principles.

While we still have massive amounts of data to analyze from both the surveys and the interviews, our studies point to the need of a deeper revision of the instructional practices for the teaching and learning of computational competencies, particularly at the introductory level to help students develop a cohesive computational knowledge based on computing principles that is well integrated with the engineering practice. Principally, it is very important to develop valid and reliable assessment instruments for pedagogical or research purposes. We will build on our existing assessment framework to refine the design and further develop performance-based assessment tools (formative and summative) and scoring rubrics to measure computational competencies for engineers.

Acknowledgments

We would like to especially acknowledge the participation and collaboration of the faculty teaching the target courses; they have been instrumental during the implementation of the project. This material is based upon work supported by the National Science Foundation (NSF) under awards 722221 and 0939065. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the NSF.

References

[1] Bernold, L. E., Spurlin, J. E., & Anson, C. M. (2007). Understanding our students: A longitudinal study of success and failure in engineering with implications for increased retention. Journal of Engineering Education, 96 (3), 263-274.

[2] Denning, P. J. (1992). Educating a new engineer. Communications of the ACM, 35 (12), 82-97.

[3] Froyd, J. E., & Ohland, M. W. (2005). Integrated engineering curricula. Journal of Engineering Education, 94 (1), 147-164.

[4] National Research Council (Ed.). (2002). Evaluating and improving undergraduate teaching in science, technology, engineering, and mathematics . Washington, DC: National Academies Press.

[5] Seymour, E., & Hewitt, N. M. (1997). Talking about leaving: why undergraduates leave the sciences. Boulder, Colo.: Westview Press.

[6] Sheppard, S. D., Macatangay, K., Colby, A., & Sullivan, W. M. (2008). Educating engineers: Designing for the future of the field : Jossey-Bass.

[7] Heldrich Center, Harvard University's Labor and Worklife Program, & National Bureau of Economic Research, 2008

[8] Alexander, P. A. (2003). The development of expertise: The journey from acclimation to proficiency. Educational Researcher, 32(8), 10-14.

[9] Bransford, J. (Ed.). (2000). How people learn brain, mind, experience, and school (Expanded ed.). Washington, D.C.: National Academy Press.

[10] Chandler, P., & Sweller, J. (1991). Cognitive load theory and the format of instruction. Cognition and Instruction, 8 (4), 292-332.

[11] Denzine, G. (2007, June). Five misconceptions about engineering students' motivation that affect the teaching and learning process. Paper presented at the American Society for Engineering Education, Honolulu, Hawaii.

[12] Smith, K. A. (1999, October 28-29, 1999). Cooperative learning and the new paradigm for engineering education. Paper presented at the ABET Annual Meeting, Baltimore, MD.

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

[14] Vergara, C. E., Urban-Lurain, M., Dresen, C., Coxen, T., MacFarlane, T., Frazier, K., et al. (2009b). 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.

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

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

[17] Vergara, C. E., Urban-Lurain, M., Dresen, C., T., Frazier, K., et al. (2011). Computational Expertise in Engineering: Aligning Workforce Computing Needs with Computer Science Concepts. ASEE, Vancouver BC, AC 2011-1050.

[18] Ohmann, P. R., Green, A. S., & Johnston, M. E. (2008). Infusing computation throughout the undergraduate curriculum. Computers in Education Journal, 18 (3), 12-22.

[19] Vergara, C. E., Briedis, D., Buch, N., Esfahanian, A-H., Sticklen, J., et al. (2012). Integrating computation across engineering curricula: Preliminary impact on students. Paper presented at the Frontiers in Education, Seattle, WA.

[20] Urban-Lurain, Mark, Anderson, Charles W., Parker, Joyce, & Richmond, Gail. (2006, March 20-24). Fluency with information technology in teacher education: Moving from novice towards expertise. Paper presented at the Society for Information Technology & Teacher Education, Orlando, FL.