Towards a Framework for Assessing Computational Competencies for Engineering Undergraduate 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.

Mr. Michael Cavanaugh, Michigan State University
Dr. Subashini Nagendran Sivakumar, Michigan State University

Suba Nagendran Sivakumar is a Research Scientist in The Center for Engineering Education Research (CEER). She received her PhD in Plant Pathology from Michigan State University. Her scholarly interests include: research and teaching in Plant Pathology, Molecular Biology and improvement of STEM teaching and learning processes in higher education.

Dr. Daina Briedis, Michigan State University

DAINA BRIEDIS is a faculty member in the Department of Chemical Engineering and Materials Science at Michigan State University and Assistant Dean for Student Advancement and Program Assessment in the College of Engineering. Dr. Briedis is involved in several areas of education research including student retention, curriculum redesign, and the use of technology in the classroom. She has been involved in NSF-funded research 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 and of the AIChE.

Thomas David Dionise, Michigan State University
Prof. 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 where he is currently serving as the Associate Chair. He was awarded 'The 1998 Withrow Exceptional Service Award', and 'The 2005 Withrow Teaching Excellence Award'. Dr. Esfahanian has over 120 research papers including articles in journals such as IEEE Transactions, NETWORKS, Discrete Applied Mathematics, Graph Theory, and Parallel and Distributed Computing. He has developed a number of software packages for manipulation of graphs. 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 data mining.

Dr. Jon Sticklen, Michigan Technological University

Jon Sticklen is the chairperson of the Engineering Fundamentals Department, Michigan Technological University. In the decade of the 90s, Dr. Sticklen founded and led a computer science laboratory in knowledge-based systems in the College of Engineering, Michigan State University that focused on task specific approaches to problem solving, better known as expert systems. Over the last fifteen years, Dr. Sticklen has pursued engineering education research focused on early engineering with an emphasis on hybrid course design and problem-based learning. Dr. Sticklen assumed the chairperson of Engineering

Fundamentals at Michigan Tech on August 1, 2014. His research has been supported by a number of companies, as well as by NSF/CISE, NSF/DUE. and DARPA. Specifically his research in DBER-based engineering education has been supported by NSF/DUE and NSF/CISE.

Dr. Mark Urban-Lurain, Michigan State University

Mark Urban-Lurain is an Associate Professor and Interim 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.

Towards a Framework for Assessing Computational Competencies for Engineering Undergraduate Students

Abstract

Assessment is a central component of educational reform. Valid assessments are key to measuring student learning and further establish a relationship between instruction and learning outcomes. Despite the central role that computation plays in engineering there remains an absence of valid assessments to measure computational competencies for engineers. The focus of this paper is to describe our progress towards characterizing students' skills and behaviors associated with computational competency as they solve engineering problems. We will describe our efforts to define a triangulation process [across data sources] that will allow us to inform the core research question: What are the features that broadly characterize the knowledge, skills and behaviors associated with computational competencies for undergraduate engineering students?

1. Introduction and Background

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. The ability of engineers to understand both engineering and computational principles which allows them to select and use computational tools to solve engineering problems by moving between physical, chemical or biological systems and abstractions in computing is crucial to the engineering practice [1, 2, 3]. Our educational challenge is to implement evidence-based instructional strategies that help engineering students move from novice understanding of computing towards competence, while maintaining motivation grounded in their engineering disciplines. To tackle this challenge and develop innovative instructional practices it is necessary to develop the foundational knowledge that uniquely characterizes computational problem-solving behavior and expertise in engineering practice.

Assessment is a central component of educational reform efforts; the use of valid and accurate assessment tools enables us to determine the impact of a given intervention on student learning and allows us to see the relationship between instruction and student learning outcomes. Assessment quickly became a central theme in engineering education and there are several examples of assessment tools and frameworks aimed to determine what a student knows and can do in the context of the engineering practice [4]. A large body of literature refers to characterizing and assessing engineering design [5, 6, 7]. However, despite the central role that computation plays in engineering practice there is a distinct lack of valid assessments to accurately measure computational competencies in the engineering context.

This paper describes our progress towards characterizing students' skills and behaviors associated with computational competency as they solve engineering problems. This work will contribute to our understanding about computational thinking in engineering and bolster the small body of research devoted to assessing computation in engineering education.

2. Project Overview

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 redesign the role of computing within engineering programs at MSU and LCC. The project comprised two phases: *CPACE I*: a) Based on employer interviews and employee surveys conducted across a representative sample of engineering businesses and industries we identified the computational competencies needed in the engineering workplace; b) To translate our research findings into fundamental CS concepts that can be used in curricular implementation we evaluated three different computational frameworks and developed a 'data-to-computer science (CS)-concept map'. Figure 1 shows the distribution of the computational competencies mapped to CS concepts i.e. *CPACE Computational Competencies*. Our results are consistent with other research on engineering education and details of the process and findings from CPACE I are presented elsewhere [1, 8-11].

In *CPACE II* our goal was to infuse computational problem-solving competencies throughout the curricula. To achieve this goal and bring about an integrated computing experience, our strategy entailed using problems derived from contemporary industrial engineering practice. We targeted courses across all four years of the engineering curricula in Chemical and Civil Engineering. Following a mixed methods approach, quantitative and qualitative data were collected such as, student interviews, surveys at the beginning and end of target courses, along with standard class data on learning outcomes, and student artifacts e.g. final project reports and homework assignments.

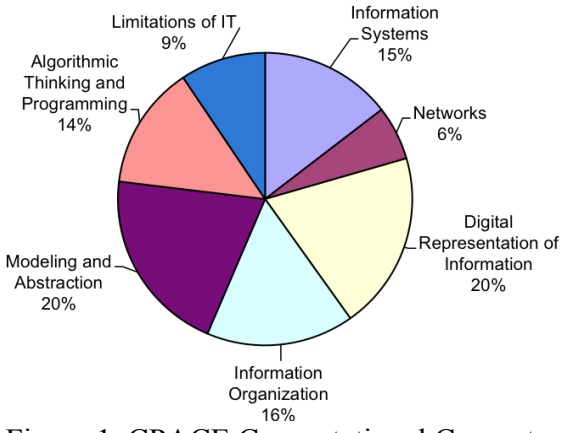

Figure 1. CPACE Computational Competencies. Distribution of engineering workplace computational competencies aligned to computer science concepts.

3. Characterizing Computational Competencies in Engineering

3.1. Computational Expertise in Engineering

Novice understanding of computing focuses on the superficial operation of particular software; expertise is grounded in a deeper conceptual understanding. A novice learns a collection of discrete skills, among which he/she sees no connections or relationships. As a novice acquires

discrete skills, he/she begins to move to the acclimation level, constructing a conceptual framework where some skills are connected to each other. Knowledge at this stage is limited, and fragmented; this fragmentation hinders learners' ability to differentiate between accurate or inaccurate and relevant from irrelevant information. Students at this stage often engage in the use of surface-level strategies that are task [assignment] oriented, and focus on discrete elements and memorization of information and techniques rather than in conceptual understanding and integration [12]. As a learner moves along the continuum towards competence and proficiency, he/she recognizes the deeper connections among the concepts of a discipline and can transfer that conceptual understanding to solving problems that require new skills [13, 14].

However, as Alexander (2003) points out, expertise is domain-specific [12, 15]. What is relevant about a given computational concept for a computer scientist is far from relevant for a mechanical or chemical engineer attempting to use the concept in practice. Our work in CPACE I revealed the same point; in order to translate our findings −computational competencies/needs in the engineering workplace —into fundamental computer science (CS) concepts we examined frameworks from computer science [16, 17] but found that they focus on principles and concepts that reflect the deep understanding of expert computer scientists [9]. The need remains to better characterize the computational competencies as applied in the context of the engineering practice.

A major effort during CPACE II —and the subject of this paper—is to determine students' computational skills and capabilities while solving engineering problems. The guiding research question is: what are the features that broadly characterize the knowledge, skills and behaviors associated with computational competencies for undergraduate engineering students?

A major challenge emerged during our initial analyses of student artifacts using the CPACE computational competencies framework (Table 1). We discovered that using the artifacts alone it was very difficult to recognize students' computational problem solving processes (CPS) [let alone measure computational competencies] as they solved engineering problems. To complement our analyses and understand the complexity of the students' computational thinking processes we conducted semi-structured interviews with the students. This paper focuses on the preliminary analyses of students' interviews and will describe our efforts to define a triangulation process [across data sources] that will allow us to characterize students' skills and behaviors associated with computational competency as they solve engineering problems. Detailed survey findings are published elsewhere [18, 19].

3.2. CPACE Computational Competencies Framework

Based on the CPACE computational competencies (Figure 1) we designed a framework to assess students' performance in course projects and other assignments as it relates to computational competencies. A team including chemical and civil engineers, computer scientists, and educational researchers developed the framework. Table I illustrates three exemplar competencies from the distribution depicted in Figure 1: Modeling & Abstraction, Algorithmic Thinking & Programming and Limitations of Information Technology. For each competency we show two sample descriptors each aligned to the characteristics and behaviors associated with the competencies.

Table 1. CPACE Computational Competencies

Modeling & Abstraction	
Descriptor	Characteristics and associated behaviors
Representation of a problem (equation, graph, relationship, simulation model)	Use of appropriate assumptions and justifications; Parsimony; Use of correct approximations; Correct choice of variables.
Abstraction of computation	Appropriate decomposition of the implementation into small cohesive parts, each tested and debugged until complete. Assembles and tests components for complete solution.
Algorithmic Thinking & Programming	
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.
Limitations of Information Technology	
Estimating of inputs and outputs	Estimating of inputs and outputs: Displays a sense of the problem scale or order-of-magnitude (e.g., length, time, other physical properties/constraints) Also includes checking the inputs provided to the models for reasonableness.
Cross-checking work.	Cross-checking work : Uses alternative methods to compare results (e.g., hand calculations, using excel and MATLAB).

4. Methods and Data Analyses

To better understand the complexity of the students' computational problem solving processes we conducted semi-structured interviews; the objective was to focus on the role of computation during the problem solving process. Drawing from qualitative methods our analyses follow three major steps common to grounded theory: coding the text for themes, linking themes into theoretical models, and displaying and validating the models [20].

4.1. Interviews

Our objectives were to better understand the process by which students solve engineering problems using computational tools; specifically how, when, and why computational tools were employed during the problem solving process. We targeted courses in both chemical and civil engineering disciplines including sophomore, junior and senior levels. We conducted a total of 13 face to face one hour semi-structured interviews, including three pilots. Operationally, students enrolled in the target courses were recruited via e-mail within two months of project completion and offered a monetary incentive. A copy of their project report was provided in advance of the interview. To prepare for the interview, the interviewers--part of the CPACE team-- reviewed the project specifications and the students' report. The interview protocol was designed to probe for three main themes: (1) general questions about the project assignment and its objectives; (2) computational themes, including probes to elicit learning and application of computational tools to problem solving and (3) probes to determine conceptual understanding of computation and the use of computational tools. The last set of questions targeted students curricular experiences as related to computation. The protocol was piloted three times to refine the final instrument.

During the interview students described their thought process as they recalled how they solved the engineering problem; a video recorder focused on the project report captured references to figures and other relevant data. ATLAS.ti qualitative software was utilized to organize and analyze the interview data. Interviews were organized according to course level; verbatim transcriptions were associated to audio so researchers had the ability to concurrently read and listen to interview data as they segmented and coded.

4.1.1. Interview Coding and Alignment to CPACE Computational Competencies

The overall process is summarized in figure 2. The boxes represent distinct steps in the process, which are detailed in the following sections. Following an immersion process, the researcher listens, reads and watches all recorded interviews taking observational notes.

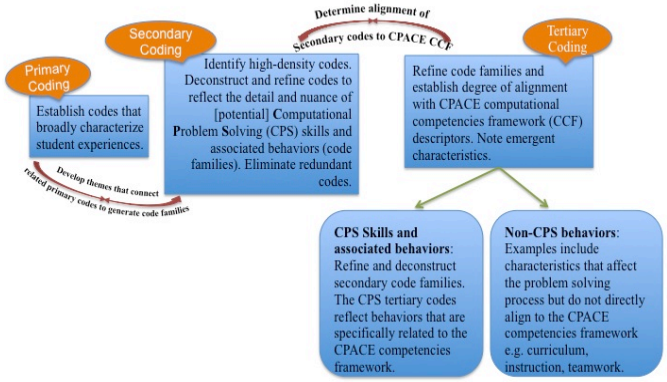

Figure 2. CPACE Qualitative Process: interview coding and alignment to CPACE computational competencies.

Primary Coding: Each segment of the transcript is coded according to the behavior exhibited. Primary codes and code definitions are established based on the utterances that broadly characterize student experiences across interviews. As one codes across interviews, patterns begin to emerge allowing for categorization and grouping. For example, a student talking about understanding a computational tool comes to define 'comprehension of tool' (COT) primary code. Each primary code has a rule of inclusion or definition that limits or allows its application. This code is then applied across interviews each time a participant exhibits 'comprehension of tool'.

Secondary Coding: Once exhaustive primary coding is completed across interviews, the researcher has a codebook with associated definitions or rules of inclusion for each code. Based on frequencies of occurrence, patterns, and co-occurrences primary codes are grouped into code families because they share some characteristic. Second order coding and theme development is focused on refinement of high-density primary codes into more strictly defined secondary codes, which include related secondary sub-codes that capture specific skills and behaviors. Thus allowing the coders to determine alignment or non-alignment with the CPACE computational competencies framework. Figure 3 depicts an example of the primary and secondary code structure; it includes the definitions for each of the codes as well as the interview quotes associated with those codes.

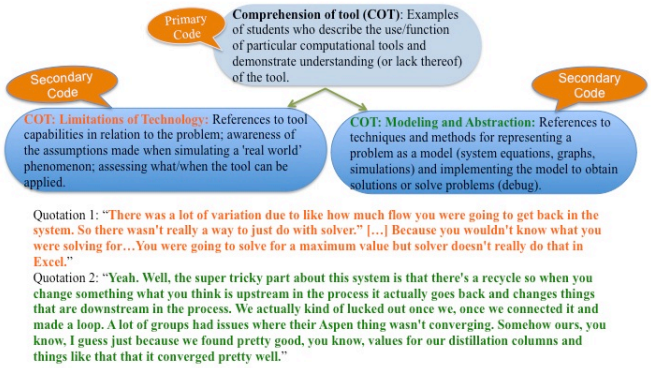

Figure 3. CPACE primary and secondary code structure: Alignment between secondary codes interview quotations are color-coded.

Tertiary Coding: After primary and secondary code/family refinement, the researchers targeted assessing code alignment to the strict constraints of the CPACE computational competencies framework. The tertiary code structure correlated more specifically to the characteristics and behaviors encompassed in the CPACE computational competencies framework. However it is important to note that using the framework as a guide did not restrict the development of emergent themes nor of additional computational competencies that might not be included in the initial framework.

Two major categories emerged from these preliminary analyses: (1) Computational Problem Solving (CPS) and (2) Non-CPS.

Computational Problem Solving (CPS): Includes all code families that reflect behaviors specifically related to the CPACE competencies framework or were otherwise related in definition or content to computation.

Table 2 depicts examples of CPS tertiary coding. It shows the alignment of the interview quotations to descriptors used in the CPACE computational competencies framework. Each exemplar tertiary code referenced in the table includes the definition, which is based on the CPACE competencies framework (Table 1).

Table 2. CPS Tertiary Code Structure

Interview Quotation	Tertiary Code Structure*
"Excel is the easiest way to do this…Because of more fast (ph) calculations […]Yeah, and we had like - the three columns they had like heat exchangers to heat the column, and condensers to cool the products… We used, and like - we had like two project - the heat exchanger. We did that in Excel, too. Like you've got the area, how much area of heat exchanger did we need, how much material, the size of the exchanger. We did like everything, those details, in Excel, yeah […] All right, yeah, that works. That was most of it. And then we got like the final result; that's the profit and the purity of the product, like the overall [..]There's the cost of the	Algorithmic Thinking & programming: Selects computational tool (programming language, software functions or uses data structures (record, array, list). Limitation of Technology: Estimating of inputs and outputs: Displays a sense of the problem scale or order-of-magnitude (e.g., length, time, other physical properties/constraints). Also includes checking the inputs provided to the models for reasonableness.

Interview Quotation	Tertiary Code Structure*
columns, the condensers. We roll (ph) their installation costs and the profit per year, how much money we've made per kilograms of product, and the purity of the product"	
"If you start looking at... Let's see... starts where like our hand calculations start coming into factor. So we started, with the furnace to figure out... He gave us an initial temperature that goes into the reactor. […] " And the two streams connect and go to a furnace in order to get up to that temperature. So then we had to calculate our heat duty. So that was like the big first step like to try to start this process is you need to get a heat duty and then you need to size your furnace using a table in our design book and then you could cost that. So that was basically like the first thing."	Limitation of Technology: Estimating of inputs and outputs: Displays a sense of the problem scale or order-of-magnitude (e.g., length, time, other physical properties/constraints) Also includes checking the inputs provided to the models for reasonableness.
"So there was actually some stuff that we needed to change and it was a lot easier to change that kind of stuff after you kind of knew what was going on in the process." […] "So, you're like, "OK. The phase separator's letting out liquid. That's not ideal. So we're going to have to add another distillation column in order to get all the stuff that we thought was going originally, in the ideal case, be already gone. We had to get rid of that stuff. So it was kind of like a thought process inside Aspen before. But you couldn't attack it without knowing what was going on what was going on in the process before."	Modeling &Abstraction: Analyze results/solutions: result is correct and recognized as such; self corrected if necessary. Meets specifications codes or constraints.
"And doing this calculation in Excel kind of helped me see what the numbers should be because I could see what numbers I put in. I understood the equation and where they came from so I can confident and say that it's at least close. So that when I plug into ASPEN - since ASPEN does a lot more complicated calculations and a lot more behind the scenes sort of thing - I want to be confident that the number it gave me was reasonable."	Algorithmic thinking & programming: Selects computational tool (Programming language, software, functions or features): knows a range of tools and is able to select the most appropriate from them to implement the solution. Limitation of Technology: Crosschecking work: Uses alternative methods to compare results (e.g., hand calculations, using excel and MATLAB). Modeling & Abstraction: Analyze results/solutions: result is correct and recognized as such; self corrected if necessary. Meets specifications codes or constraints.

*Color-coding is used to indicate code alignment to the interview data.

Non-Computational Problem Solving (Non-CPS): Includes characteristics and behaviors that affect the problem solving process but do not directly align to the CPACE competencies framework and are not specifically computational in nature. These code families reflect a depth of additional information related to actual student computational problem solving processes, for example these code families identify several emergent themes that might further inform questions centered on the role that sociocultural and disciplinary factors play during problem solving within the engineering context. Table 3 shows examples and definition of Non-CPS code families. Table 4 shows examples of the codes aligned to specific interview quotations.

Table 3. Non-CPS Code Families and Definitions

Non-CPS codes/categories	Definition
Self awareness	All examples of student self-awareness...this might include all references to student awareness of strengths and limitations when working towards solutions to problems or with particular tools.
Social Impact-engineering culture	All examples of respondents who refer to 'most people' or on behalf of the larger community that he/she is a part of...students might relate how, most 'engineers' or 'engineering students' think or act.
Social Impact: validating/learning	Any example of respondents recalling a social intervention or interaction that aided or detracted from learning/problem solving. Students might reference validating/problem solving with other groups, mass consensus, lab interactions, comparing solutions/approaches, outside resources, and individual/pair programming strategies.
Teamwork	All references to collaboration, which might include working in a team, group, or partnership while solving problems.
Research Role-integrating resources-computer and human	Any reference to student research role and understanding of place in service of problem solution. All references for students seeking and integrating resources computer and human.
Assessment	All references to the evaluation or estimation of the nature, quality, or ability of someone or something. Examples might include assessment of professors, courses, tools, projects, and other students.
Interviewer confirmation	All references to interviewer needing confirmation of problem content, process, or objectives. This code might also refer to any point at which the interviewee affirms or reacts to interviewer probes meant to facilitate dialogue on process but only results in single word answers.
Adaptability/learning	Instances where students find resources (web resources, additional computational tools, books, manuals, teachers, peers) to help solve problems/complex questions. Evidence student sought answers in diverse places and applied them.

Table 4. Non-CPS Code Structure

Non-CPS codes/categories	Interview Quotation Exemplar
Self awareness	R: So I think it was kind of, you know, I know my own limitations. I know that if I just go through and try to run through it all, I'll wind up, you know, messing something fairly simple up. R: I think it's kind of too because everything was connected. I mean, you couldn't really move onto the next block in the whole design until you knew what was going to be input from the last block so it was like, "Okay, we have to work through this together to kind of make the process work." Otherwise, if I have to try to jump ahead three steps and start there, I'm never going to know if that's what I should have started with.
Social Impact-engineering culture	R: I think you would get, you wouldn't get the same exact answer but you would get something relatively close. RESPONDENT 1: Yeah. Yeah, for our project, I don't think, we won't be able to do that but maybe someone more experienced than us. [01:06:57] R: Like a ten-year chemical engineer that understands everything a little bit more. Someone with like five, ten years experience in the field would actually know exactly what was going on so that they would have a little bit better say than somebody that's just going through it right now. I mean, like we said, we didn't

	really have as much experience with this stuff until we actually got into the process.
Social Impact: validating/learning	R: Oh, no, seniors came in to our junior class and did the tutorials that we did this year. Yeah. I think it's just, it seems like there's just not a lot of time to like do a lot of explicit instruction. We definitely use the polymath in reactions. We run over like the outputs and everything. I think maybe I just didn't understand that as well as like some other people. I know that's definitely the case. But, I think it's also like my learning style too. Like I know a lot of people in our class learn by doing and learn by examples and that's what we got for all this stuff. We got great examples and like do it and figure it out and I guess I'm just a little different in that way
Teamwork	R: "…Aspen's pretty, it's pretty tough to find help on Aspen online… It's pretty much like relying on other students to, you know, like, "Hey, how did you model this because I, it's not working for me. Like, "Oh, you know, we used, we used this particular kind of distillation column or, you know, this kind of heat exchanger as opposed to this one." So it was kind of just guess and like, I don't know, guess and check for ourselves but kind of just trying out things for ourselves and then confirm with other groups to see what they did"
Research Role-integrating resources-computer and human	R: "It's pretty tough to find help for Aspen online because you can never really find what you're looking for. I mean, you know, you can Google like, you know, Aspen distillation columns but it won't be the same system that you're using so it's not really going to be helpful. Aspen's pretty, it's pretty tough to find help on Aspen online"
Assessment	R: A lot of stuff in some of these programs is really just not formal education, but learning by doing. So like I said, with the headings, it's just someone else showed me how to do that and I learned how to reference by doing it in Microsoft, and I feel like that might be a skill that is good to talk on, if something's like for like, in things of Microsoft, sometimes you can teach yourself by like doing it or by like Googling, like, if you like think there should be a way to do this in Excel, like if I made this programming, I would do that"…So a lot of things, I think, are just learning by doing, or like having that ability to know that there is a way to do that easier…"
Adaptability/learning	R: ...in order to get something more like your ideal case. So, we couldn't... We tried a couple different methods. We tried a base separator in order to get out the methane that was coming from the phase separator in order to get the methane that was coming out of the phase separator. That didn't seem to work very well. There was still a little bit... R: That was our original way of trying to figure it out and then we realized that that wasn't really working for us. So then we, we decided to step it up and the next step from phase separator is like a small distillation column.
Interviewer confirmation	I: And when you say calculated by hand, you're saying literally by hand with a calculator, using a spreadsheet? R: We used a, pretty sure we used a spreadsheet for this too just because it was, it was easier than doing it, like actually writing it out by hand. R: Yeah. I think we did use a spreadsheet. I: Right. But again, it's basically a sort of thing you could use a calculator for [...]"

R= Respondent; I=Interviewer

Inter-coder Reliability and Arbitration: inter-coder reliability tests were performed after primary and secondary coding to ensure that multiple coders perceived the same constructs/codes applied to the same interview data segments. Investigators including chemical and civil engineers as well as computer scientists were given the codebook and a subset of interviews and asked to independently code. While final analysis is still pending (but will be included in the final manuscript) preliminary results indicate robust coding consensus. Furthermore, the exemplars used in this draft include a second round of inter-coder testing by a member of our team with expertise in both computational science and educational research methods.

At the time of writing we have completed primary and secondary coding for all the interviews (13 total) representing chemical and civil engineering disciplines and including sophomore, junior and senior level courses. We also completed most of the tertiary coding for all interviews from chemical engineering including sophomore, junior and senior level courses and interviews from civil engineering including sophomore and junior level courses. The interview quotes used in figures and tables are taken from the chemical engineering interviews. Due to space restrictions tertiary-coding exemplars presented here are focused on three CPACE computational competencies: Algorithmic Thinking & Programing, Modeling & Abstraction, and Limitations of information Technology.

5. Summary and Discussion

The lack of assessment instruments to measure engineering students' computational competencies represents a major challenge in the implementation of the CPACE project. To address this challenge, the CPACE team defined a process that allows for rigorous analysis and triangulation across data sources as a way to address the perceived gap in our understanding of the role that computation plays in students' actual problem solving processes in an engineering context. In this paper we present our efforts to define the triangulation process and show preliminary analyses of the students' interviews.

Our initial approach involved using a set of rubrics based on the CPACE computational competencies framework (Table 1), to compare self-reported survey data regarding students' computational abilities and actual performance in classroom assignments. This approach made it very difficult to recognize students' computational problem solving processes (CPS) as they solved engineering problems. To complement our analysis we used qualitative interview data along with survey data and student artifacts. The developed interview codification process and analysis was grounded in the lived experiences of students, which allowed for a deeper examination of the role computation plays during problem solving. In addition, the qualitative process helped the CPACE team identify several emergent themes that might further inform rubric development and future research questions centered on the role that sociocultural and disciplinary factors play during problem solving within the engineering context.

The examples presented here represent only a fraction of our qualitative analyses and are for the purposes of demonstrating our process of creating a framework for assessing computational competencies for undergraduate engineering students. However, the exemplars do point to preliminary understandings of the important role computation plays as students solve complex engineering problems. We highlight two broad categories, Computational Problem Solving (CPS) and Non-CPS, each encompassing several related code families that expose a more focused view of how students approach computational problem solving and operationalize knowledge. Currently, we are completing the categorization, coding process and alignment to the CPACE computational competencies framework.

Initial analyses reveal alignment with survey findings; a conclusion of our study is that students have difficulties making connections between different computational tools. Moreover, students have difficulties utilizing learned computational skills in ways that are integrated and operationalized across new and different contexts. This points to a limited understanding about

underlying computational principles. 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 are well integrated with the engineering practice.

The CPACE computational competencies framework helps disaggregate the level at which students think computationally across curricular levels and thus allow researchers to address particular needs. Principally, it represents a step forward in the development of assessment tools and scoring rubrics to measure computational competencies for engineers. These instruments can be used for pedagogical and research purposes. For example, faculty can design instruction that is more appropriate to the ways that students organize their knowledge of the subject.

Challenges/Limitations:

During the interviews students were asked to describe or recount their thought process surrounding an already completed project and the time elapsed between interview and completion of the project may have impacted interview responses. Further, by the very nature of the research and interviews, leading questions and interviewer confirmation of particular computational processes were inherent, which may have introduced leading questions and biased answers. Moderator acceptance bias may have also been present, whereby interviewees provide answers to please the moderator. Respondents might interpret what they believe the moderator wants to hear and answer accordingly. All instances of bias were noted during coding process.

Acknowledgment

This material is based upon work supported by the National Science Foundation (NSF) under award 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] 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] Sheppard, S. D., Macatangay, K., Colby, A., Sullivan, W. M. & Shulman, L. S. (2008). Educating engineers: Designing for the future of the field: Jossey-Bass.

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

[4] Olds, B. M., Moskal, B.M. & Miller, R. L. (2005). Assessment in Engineering Education: Evolution, Approaches and Future Collaborations. Journal of Engineering Education; 94 (1), 13-25.

[5] Kilgore D., Atman C.J., Yasuhara K., Barker, T.J. & Morozov, A. (2007). Considering Context: A Study of First-Year Engineering Students. Journal of Engineering Education 96 (4), 321-334.

[6] Adams, R. S. & Atman, C. J. (2000). Characterizing engineering student design processes: An illustration of iteration. Proceedings, ASEE Conference and exhibition, 2000.

[7] Atman C., J., Adams, R., Cardella, M.E., Turns, J., Mosborg S. & Saleem J. (2007). Engineering Design Processes: A Comparison of Students and Expert Practitioners Journal of Engineering Education October 2007, Vol 96(4), 359-379.

[8] Vergara, C. E., Urban-Lurain, M., Dresen, C., Coxen, T., MacFarlane, T., Frazier, K., et al. (2009). 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.
[9] 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.
[10] Committee on the Engineer of 2020, Educating the engineer of 2020: Adapting engineering education to the new century. National Academy Press: Washington, DC, 2005.
[11] Educating Engineers: Designing for the future of the field. The Carnegie Foundation for the Advancement of Teaching 2008.
[12] Alexander, P. A. (2003). The development of expertise: The journey from acclimation to proficiency. Educational Researcher, 32(8), 10-14.
[13] Bransford, J. (Ed.). (2000). How people learn brain, mind, experience, and school (Expanded ed.). Washington, D.C.: National Academy Press.
[14] Byrnes, J.P., (1996). Cognitive Development and Learning in Instructional Contexts, Boston, Mass.: Allyn and Bacon.
[15] Alexander, P. A., & P. K. Murphy. (1999). Nurturing the seeds of transfer: A domain-specific perspective. International Journal of Education Research 31:561–76.
[16] Denning, P. J. (2003). Great principles of computing. Communications of the ACM, 46(11), 15-20.
[17] Wing, J. M. (2006). Computational thinking. Communications of the ACM, 49(3), 33-35.
[18] Vergara, C. E., Briedis D., Buch N., Esfahanian, A-H., Sticklen, J., Urban-Lurain, M., Paquette, L., Dresen, C. & Frazier, K. (2012). Integrating Computation Across Engineering Curricula: Preliminary Impact on Students. FIE Annual Conference. Seattle, WA 2012.
[19] Vergara, C. E., Urban-Lurain, M., Sticklen, J., Esfahanian, A-H., McQuade, H., League, A., Bush, C. J. & Cavanaugh, M. (2014). Towards Improving Computational Competencies for Undergraduate Engineering Students. Paper to be presented at the ASEE annual conference Indianapolis, IN, June 2014.
[20] Bernard, Russell H. (2011) Research Methods in Anthropology: Qualitative and Quantitative Approaches-Fifth Edition, Plymouth, United Kingdom. AltaMira Press.