The Alignment Between Formal Education and Software Design Professionals' Needs in Industry: Faculty Perception

Secil Caskurlu, Purdue University

Secil Caskurlu is a doctoral candisate in the Department of Curriculum and Instruction, Learning Design and Technology program at Purdue University.

Ms. Iryna Ashby, Purdue University

Iryna Ashby is a Ph.D student in the Learning Design and Technology Program at Purdue University with the research interests focused on program evaluation and competency-based education. She is also part of the program evaluation team for the Transdisciplinary Studies in Technology – a new initiative at Purdue Polytechnic aimed to redesign undergraduate student experiences through offering a combination of deep liberal arts experiences with student-driven, hands-on project-based learning.

Dr. Marisa Exter, Purdue University

Marisa Exter is an Assistant Professor of Learning Design and Technology in the College of Education at Purdue University. Dr. Exter's research aims to provide recommendations to improve or enhance university-level design and technology programs (such as Instructional Design, Computer Science, and Engineering). Some of her previous research has focused on software designers' formal and non-formal educational experiences and use of precedent materials, and experienced instructional designers' beliefs about design character. These studies have highlighted the importance of cross-disciplinary skills and student engagement in large-scale, real-world projects.

Dr. Exter currently leads an effort to evaluate a new multidisciplinary degree program which provides both liberal arts and technical content through competency-based experiential learning.

The Alignment between Formal Education and Software Design Professionals' Needs in Industry: Faculty Perception

Secil Caskurlu, Iryna Ashby, & Marisa Exter

Purdue University

Abstract

Given the competitive global market for computing professionals and increased undergraduate enrollment rates in computing majors in the US, it is important to continually review how computing education programs prepare their graduates for success after graduation. This study aims to explore faculty perceptions about the degree to which computing-related programs meet the needs of software design professionals in the job.

The Alignment between Formal Education and Software Design Professionals' Needs in Industry: Faculty Perception

Introduction

Given the increased undergraduate enrollment rates in computing majors among US computer science departments[1], it continues to be important to consider how computing education programs prepare their students for success after graduation. The advent of the knowledge economy, globalization, and rapidly developing technology make a recent undergraduate possessing only discipline-specific skills unprepared to address the types of ill-structured problems they will encounter in the work-place[2-4]. According to a survey conducted by the Association of American Colleges and Universities (AACU), employers ranked the following skills as important for graduates: clear written and oral communication; the ability to solve complex problems in a real-world setting; the ability to demonstrate ethical judgment and integrity; intercultural skills; the capacity for continued learning, and general breadth of skills and knowledge [5]. Thus, having disciplinary knowledge is not enough. Today's workplace requires applying such knowledge towards analysis, decision-making, and problem solving within a complex environment[6-8].

It is therefore unsurprising that, in the Computing Science Curricula 2013 final report, the Joint Task Force on Computing Curricula[9] stated that

> The education that undergraduates in computer science receive must adequately prepare them for the workforce in a more holistic way than simply conveying technical facts. Indeed, soft skills (such as teamwork, verbal and written communication, time management, problem solving, and flexibility) and personal attributes (such as risk tolerance, collegiality, patience, work ethic, identification of opportunity, sense of social responsibility, and appreciation for diversity) play a critical role in the workplace. Successfully applying technical knowledge in practice often requires an ability to tolerate ambiguity and to negotiate and work well with others from different backgrounds and disciplines. These overarching considerations are important for promoting successful professional practice in a variety of career paths. (p. 15)

Consistent with the Joint Task Force on Computing Curricula, the Accreditation Board for Engineering and Technology (ABET)[10] also highlighted the following skills required across engineering and technology programs along with technical skills: effective communication; functioning on multidisciplinary teams; problem solving; understanding the impact of solutions in global, economic, environmental, and social contexts; and lifelong learning. In a recent mixed-method study, Caskurlu, Exter, & Ashby (2016) [11] found that computing professionals from various industries believe that problem solving, critical thinking, lifelong learning, teamwork, and interpersonal skills are as important as technical and industry-specific skills. However, computing professionals who graduated from computing related programs within the last five years indicated that lifelong learning, teamwork, interpersonal, and communication skills are inconsistently emphasized in formal education. Consistent with Caskurlu and Exter (n.d.) [12],

previous research focusing on the alignment between formal education and needs of the industry also indicated that technical programs should look beyond single-domain skills and bridge disciplines that were once considered separate [11, 13-16]. This would help programs to incorporate non-technical knowledge and skills, such as communications and interpersonal skills, business strategy and applications, team/organization and management, general problem solving and critical thinking, as well as self-regulated lifelong learning skills[3, 4, 14, 16-18]. Employers report that employees are not well prepared in terms of global knowledge, writing, critical thinking, adaptability, self-knowledge, oral communication, or quantitative reasoning[19]. However, the question remains: how to effectively design the curriculum to ensure that both discipline-specific and cross-disciplinary skills are sufficiently covered? To address this question, we need to explore the issue from multiple angles: (1) gaps in education perceived by faculty, (2) gaps in education perceived by employers, and (3) gaps in education perceived by recent graduates in the field. This qualitative research focuses on the first angle in order to answer the following research question:

- What are the perceptions of computing education program faculty members about the alignment between formal education and graduates' needs on the job?

Methodology

Research Design

A phenomenological approach was used to explore and understand faculty perceptions about the degree to which their computing-related programs meet the needs of software design professionals in the job. A phenomenological approach allows to focus on "a paradigm of personal knowledge and subjectivity, and emphasize the importance of personal perspective and interpretation" ([20] p. 1). Thus, a phenomenological approach gives insightful detail about

participants' individual perceptions about a specific phenomenon in order to understand their experiences. The selection of this approach was determined by the intent to explore the phenomenon (or "lived experienced," [21-23]) of the faculty teaching in such programs. The study was approved by the Institutional Review Board at Purdue University.

Participants

As suggested by the phenomenological approach[21], participants were purposefully selected based on the goal of the research, and their experiences and expertise relating to the phenomenon to be researched [24]. Eight faculty members in computing education programs, including degree programs in computer science and computer information technology, participated in this study (see Table 1).

Interviewee	Gender	Program	Teaching Experience	Title	Industry Work experience	
					Currently working	Worked Before
Olivia	Female	CS	5+	Adjunct faculty	No	No
Josh	Male	CS	25+	Emeritus Professor	Yes	NA
Michael	Male	CS	30+	Professor	No	Yes
Emily	Female	CIT		Professor	No	No
Daniel	Male	CIT	10+	Professor		Yes
George	Male	CS	10+	Associate Professor	No	No

Brian	Male	CS	25+	Emeritus Professor	Yes	NA
Lily	Female	CS	<1	Adjunct	Yes	NA

Table 1. Demographic distribution and current positions of faculty participants (n=8)

As summarized in Table 1, two are adjunct and two retired. Furthermore, two are currently working in industry as well as teaching, and one is working as an administrator at a university.

Data Sources

In-depth semi-structured interviews were used for data collection purposes. The interviews were conducted face-to-face, over the phone, or via Skype, depending on the participant's preference, and were recorded with each participant's permission. Interview questions (see Appendix A) focused on participants' perceptions about the most emphasized skills in their courses, skills required in the job market and how much these skills were stressed in their courses, to what extent current computing programs meet the needs in the job, and how their computing education programs could be improved.

Data Analysis

Provisional coding was used to analyze the data[25]. A provisional code list was formed from the interview questions. Data were coded according to the following four major themes: (1) degree to which formal education prepares students for their future employments, (2) courses extensively taught by their program and importance for the professional life, (3) important skills required by professional life and degree of coverage in formal education, (4) suggestions for ideal undergraduate programs. Subsequently, subthemes and patterns were identified within each

theme in response to the interview questions. Finally, significant statements, which provide a rich description of themes and/or patterns, were quoted as examples for each theme.

Reliability and Trustworthiness

To establish the inter-coder reliability, the coding of the interviews went through a three-phase process. First, the first author coded all the interviews individually. Then the second author went through all codes to prevent misunderstandings and/or identify uncommon coding cases. Finally, any disagreements that occurred were resolved. This process continued until 100% inter-coder reliability was reached.

Results

Level of CS Student Preparation for Employment by Formal Education

All faculty members interviewed believed that formal education provides students with the fundamental skills necessary for software development. As one participant said:

> *University gives to students the familiarity with the subject. However, real experience comes to you with the job. But after going in work life, the subjects or classes you study in school help you to understand the work requirements and eas[ily] transition to the work's projects. In other words, [a] CS department (also, all other departments) prepares the students what to expect when they step in the professional world.*

However, all but one participant agreed that even though schools prepare students well in terms of technical skills, there are areas that need further focus, such as problem solving, critical thinking, lifelong learning, teamwork, and communication with end users. For example, one faculty member, who is retired and currently has been working in industry for five years, explained:

> *In terms of understanding that software is a team sport, they don't understand this at all. They are not used to, or have a bunch of experience with how to do software in teams. They don't understand the sort of best practices in trying to make a long-term team work together, like how you communicate in ways that increase friction or decrease friction.*

Courses Extensively Taught and Their Importance for Professional Life

The courses extensively taught in computing-related degree programs vary across different programs and faculty concentrations. Yet software programming skills, algorithms, data structures, operating systems, networking, and understanding distributing systems were identified across participants as the most emphasized courses.

All participants considered these courses to be relevant to gain fundamental knowledge on how computers function in order to see the big picture of software development. For instance, one of the participants mentioned that his program designed the curriculum to prepare their graduates for professional life:

> *I hope all of our courses are getting our students ready [for] professional life. We've design the curriculum to make sure our students are prepared for the workforce, and by all accounts we've got 90% of our students without an internship get internships, our students have very high placement rates, and very high salaries for our college, and really for Purdue. The only product we have is our students, and the market had really voted [for] that product.*

One participant mentioned that since computer science departments have become very broad, courses that are extensively taught vary between departments based on the current faculty

members' backgrounds, causing coverage of the fundamentals to be uneven. As this emeritus faculty member explained,

> *Computer Science has gotten very broad. If you came here in 1970s or 80s, we knew exactly what to teach. Every student had to learn about compliers, operating systems, a little bit about numerical analysis, and some theory of algorithms. They had those four basics plus a few electives, they were golden. Computer science now is so broad, that there are entire sub-areas that I know virtually nothing about. . . . But right now we are in one of these diverting areas. We are in a time when we are just starting anew . . . in fact the faculty coming in this year, they want to have brand new things that we don't even have yet. Even more. So, are we teaching the fundamentals? Well, students who listen carefully and take the right tracks, I think they can get some real good fundamentals. But, I hate to say, students often take the easy way out. They look for the tracks with the easiest courses.*

This participant emphasized that it is imperative to track what companies look for and what students want to do after graduation.

Important Skills and Level of Focus in Formal Education

All faculty members agreed that both technical and soft skills are important to have in well-rounded professional preparations.

Technical Skills and Industry-Specific Skills

In addition to domain-specific skills, industry-specific skills also play crucial roles. For instance, one faculty member sees skills beyond development, like software testing, as the most important ones to have:

Some of the deep understanding of testing methodology is, I think, the most important skill. That is, how to verify that your implementation satisfies your specifications and so on and so forth. And also understanding of software maintenance is something that we can't possibly teach our students. But somehow we've got to get them exposed to that in a curriculum. I mean, students don't go out into industry and do the same kind of work that they do in school. The nature of the work is entirely different. So somehow if we can get them involved in rather than just have a superficial knowledge of terminology in software maintenance, that would be, I think, the most important thing we could send them out with.

Soft Skills

All faculty members agreed that soft skills, such as lifelong learning, teamwork, and communication, are as relevant for software developers as domain-specific skills.

Lifelong Learning

Faculty members agreed that lifelong learning is critical for professionals, because they cannot teach everything within the typical timeline of formal education and technology is constantly changing. As one faculty member stressed, employees look for people able to surpass "what they learn in class, because we can't teach students every possible thing they need to know. They need to understand how to put together in a new ways, open things up."

Teamwork

The vast majority of faculty highlighted the importance of being able to work in teams as part of a professional career. As one said:

[Teams are] very important. These are the problem [solving] tools, to work in a team, to understand how you can participate in a team, how you can organize your work so that it

> *works in collaboration with others. You also need to know how to lead a team. This is, for any programmer going into the professional world, it is pretty much a dead-end job, unless you understand what it is to be a leader, and to build a team, and to do more than just sit and write code. And so, this is very important for the professional development of anybody going out into the software development world, to understand how to have enough creativity and insight into the way people work, understanding what talents different people have, how to make them all work together in an effective way.*

Communication

Communication was seen as paramount for the software development world, to be able to communicate across a diverse range of recipients, from immediate team members to end users. One of the faculty participants with over 30 years of experience saw communication as being more essential than some of the skills traditionally considered to be fundamental: "interpersonal skills, communication, writing and communicating verbally, being able to participate in a team [is more important] than any ability to solve calculus problem or coding." In the same vein, another faculty member mentioned "[Y]ou have to be able to communicate effectively orally, verbally, and in written [form] with rest of your team. Otherwise you don't have any value."

Problem Solving

Problem solving and curiosity were seen as magic ingredients for a successful CS career. For example, one participant stressed the importance of problem solving and curiosity for the professional life:

> *I still think that there is one magic ingredient or two magic ingredients that we can teach; one is curiosity. . . . I need the combination of curiosity and problem solving, just jump in*

> *and do it. And if you go to industry, they find very hard. . . but that's what they are looking for.*

Degree of Coverage in the Curriculum and Their Individual Courses

When we asked about the degree of emphasis of the skills described above in the CS curriculum, all participants explained that these skills are not stated explicitly in the curriculum because of a number of reasons: (a) faculty are not trained this way, (b) faculty do not sufficiently value them, and finally, (c) faculty have a heavy curriculum load and not enough time to address or even incorporate these skills. As one participant explained,

> *This is part of team leadership, program management. It is hard to . . . here is the problem; it is hard to [do] that in a semester course/a quarter course and the project also takes a month or two. [To] do project management in such a short amount of time is a very difficult challenge unless you have an entire course on project management. In that case, probably, it would be tough for somebody other than traditional computer science faculty.*

Even though not explicitly emphasized in the curriculum, all of the participants try to integrate these skills into their courses via semester long projects, writing assignments, or asking students to solve complex problems. As one faculty member said,

> *In fact I tell students that if you want to be a good computer scientist, here is what I recommend you [do]: get an old, slow computer and think about why it is slow and what you can do to make it faster, now you got a real challenge. They write software and they measure it.*

Another faculty member mentioned that she asks students to write papers in order to improve their communication skills:

> *Communication, I think, is super critical to teach. I think that it's generally not especially part of a software engineering curriculum. . . . I require people to write papers and write quiz answers out. I didn't have ABCs. I wanted them to write short paragraphs because being able to write fluently is, I think, super critical.*

Ideal Undergraduate Program

When asked about the most valuable college/university experiences that prepare students for the job, the majority of participants (n=7) believed that, formal education provides students with the foundations of software development, including literacy in the discipline. Participants highlighted that even though soft skills are key for a professional career and ought to be to be part of the curriculum, to prepare students for future employment However, participants indicated that they felt soft skills were not sufficiently stressed in comparison with technical skills.

Potential Adjustments to Math Requirements

One faculty member suggested that programs should have less focus on calculus and more emphasis on discrete mathematics, probability, and machine learning to meet industry demands:

> *In the past, computer science majors required certain courses in mathematics, like calculus, that sort of thing. That's a total waste of your time. Every computer science student, every software developer, has to have 2-3 years of statistics and probability theory. That's critical. And that's what we don't see very often… that's something that if it's not there, they need to throw out all the requirements for calculus and that type of thing, and add statistics, probability, and data analytics.*

Liberal Arts Integrated Curriculum

All faculty members believe communication skills are very important for a professional, and two participants specifically recommended including liberal arts elective courses to help students develop those skills. One faculty member recommended rethinking the way a program is structured to leverage experiences outside of the classroom, which may serve to develop key non-technical skills.

> *I specifically would like to see little bit more choice and opportunity in how some of the basic requirements are achieved. We require those to be in a classroom right now, that are in many cases, small groups with graduate students. I wonder if we wouldn't be more successful basically having our students . . . work on communication and English as part of their summer jobs, and use some written product from jobs that can be academic projects. That might allow us to repurpose some of the credits in a way that lets students get things done in a more timely way, that just take up time in the first year or 18 months of their degree – it's part of the problem. Students don't see many things in their major that they like, so they go around and find other things they like. . . . having to apply the communications skills and English skills, in context.*

Interdisciplinary Capstone Projects

As indicated in previous sections, all faculty members think that being able to work in a team is crucial for success in the workplace. In addition to teamwork, three faculty members highlighted the importance of providing experience with working in interdisciplinary teams and projects in undergraduate courses, since the problems they face in the real world are no longer limited to individual disciplines, and it is less and less frequent that all members of a team come from the same background or play the same role. As one faculty member said:

> *We should be doing more interdisciplinary work. . . . It is important for professional life, because problems are not in disciplines, not any more. All disciplinary problems have been solved for a long time. The really interesting problems are interdisciplinary problems or transdisciplinary problems. Every computer scientist is going to be working with people from other fields. If we pretend in higher education, then we can put people into little isolation and they become educated, that is not preparing them for the reality what happens outside of the university.*

In addition to the interdisciplinary projects, it is also important to have real world collaborators and/or clients to help students develop their identities as software designers or developers. For example, one participant said, "I think allowing them and encouraging have more … outside influence and contributions [to] basically shap[e] what they think [as] an engineer or computer scientist or IT would be much valuable." Another faculty supported this idea by saying, "The one thing I would like to see [is] more applied work integrated more widely into the classes."

Soft Skills

Teamwork

All participants mentioned the importance of teamwork. They believed that along with working in teams or getting along, solving problems as part of a team is particularly crucial. One participant said, "Some interpersonal skills and teamwork are important, but I don't think we specifically trying to make that work". Another participant highlighted the need for more emphasis on teamwork skills by saying:

> *People work in teams. So one needs to learn how to work in a team and most academic programs tend emphasizing individual accomplishment in terms of programming*

projects. But there is the real need to learn as part of team to solve the problem to build a system; that's absolutely the most important thing

All faculty members affirmed that they cannot teach every necessary skill in just several semester-long courses. A graduate needs to realize the need to continue learning, as well as have the ability and desire to learn to meet and exceed the needs of their current roles in order to succeed. As one faculty member explained:

You need to be well-rounded, because we're not going to keep the very technical person who can't do all the other stuff when there's only one position. We have to have sort of a person that can do all kinds of things, if we're only going to have one of those positions. . . . Mobile didn't exist . . . what, 10 years ago? There was no mobile app development. Now, that's where we think most of the new application development [is] coming from. There has to be some element that helps people and that's why I really like self-trained people who didn't go to college; they're used to sort of bootstrapping, figuring out, "Okay, there's this new technology on the scene. What does it do? How can I use it? What can I do?" They didn't have it spoon-fed to them, and so it really helps them to be capable of changing and evolving to that new technology.

Self-evaluation

One faculty member also mentioned that graduates need to be able to evaluate gaps in their own level of knowledge and skills, which is as important as completing projects:

I think orienting students away from low-level [instruction is important]. Remembering, understanding, application, integration, and evaluation. Particularly high-level evaluation is important because some people work on projects [that require the] ability

to evaluate whether something is good or bad and we almost never ask students to do that. We ask them to work but not to evaluate.

Increasing Diversity

Diversity was viewed to be an important issue in computing programs. One faculty member mentioned that women should be encouraged more to become software developers to increase the diversity in the field:

The other thing is that the current system in the academic community does not do enough to encourage women to go into the field. . . . I am not sure what should be done about that, but I think perhaps more team development might be a good way to do it. It may require rethinking about the way the courses are done. There is a great emphasis on the academic; [in the] software development world [there] is a huge emphasis on obsessive programming skills, building software to the exclusion of everything in real life. I see that often in the professional world. That is not particularly healthy, and it does not attract a diverse base of people to this. . . . That was a part of the problem with the imbalance of gender in CS programs.

Discussion and Implications

This study aimed to explore faculty perceptions about the degree to which computing-related programs meet the needs of software design professionals in the job. As suggested in previous research and confirmed by our results, soft skills such as problem solving, teamwork, communication, and lifelong learning are crucial for graduates to become well-rounded professionals[9-12,14]. These findings are not surprising. As mentioned by some of our participants, the world of technology and computer science is constantly evolving. Even more so, products developed by computing professionals become an essential part of everyday life or even

integrate different aspects of our lives. As a result, graduates need to be able to recognize and communicate such changes, as well as understand their own gaps in knowledge and skills to be able to address such changes in the future.

Even though the faculty participants stated that they believe their students are well-prepared for a career in terms of domain-specific knowledge and skills, student preparation in terms of soft-skills may not be sufficient, since it is often only inconsistently addressed on a course-by-course level. As such, our results suggest the following two measures to further improve computing programs.

First, soft skills should be part of formal education to support not only collaborative efforts, but also an ability to address ill-structured problems that are commonly faced by professionals today. Indeed, we recommend that the concepts "soft skills" and "humanities"/"liberal arts" be looked at more carefully. Although our participants stressed the importance of communications skills, earlier studies[12-13] highlighted the importance of other skills and knowledge, and even the concept of a liberal education as a whole, which some participants indicated helped them understand the world as a whole better, leading them to become both better collaborators and better designers.

Second, life-long learning skills should be taught or modeled within course work and experienced through non-course experiences. This may require rethinking faculty expectations about the way students work together, and find and use sources, as well as assessment practices. As discussed in our earlier work[14-15], habits that may be construed as cheating within a traditional educational environment are considered not only good life-long learning practice, but are essential skills for success on the job by computing professionals, including, but not limited to, asking for help and feedback from mentors and peers; reusing existing code and components

were possible; and finding sample design or code in books, online sources, or their own prior work to experimenting with and adapting it to their own use.

Limitations and Future Research

Study participants were identified purposefully based on their background, teaching areas, and research interests. However, some of the faculty members we contacted either were not able to participate or not interested in the study. The sample group may not be representative of all computing programs or all faculty within them. For instance, participants included in this study all teach at four-year institutions, therefore community colleges and two-year colleges were not represented in the study. It is also likely that those willing and able to participate in a study of this nature are more likely to be interested in innovations in computing programs than computing faculty in general. Therefore, findings should not be taken as generalizable to the larger population, but they do highlight areas of interest for continued research, especially as these themes align well with earlier data collected from in-practice professionals. In the next stage of our research agenda, we will survey computing faculty, allowing us access to a larger and more representative faculty pool. This will also allow us to compare responses of a larger faculty pool to those of professionals reported by Exter and Caskurlu (n. d.)[12].

The primary groups who might be interested in the result of this study are researchers, educators, curriculum developers, and administrators in computing education fields. The results of this study may be of value to researchers who are interested in curriculum development in computing education programs, education of software design or development professionals, and teaching computing or engineering in context. In addition, this study may indicate ways that undergraduate programs in computing education could be improved to support the development of computing professionals and enhance the effectiveness of programs. Future studies will

explore similar issues in other design related fields (such as Instructional Design or Computer Graphics), and compare findings across fields. This may be of interest to design researchers and curriculum designers looking for common ground.

Acknowledgements

The researchers would like to thank the Purdue Research Foundation for supporting this publication and our research.

References

[1] Zweben, S. (2014). Computing Degree and Enrollment Trends. *Computing Research Association*. Retrieved from cra.org/uploads/documents/resources/taulbee/CRA_Taulbee_CS_Degrees_and_Enrollment_2012-13.pdf

[2] Besterfield-Sacre, M., Cox, M., Borrego, M., Beddoes, K., & Zhu, J. (2014). Changing engineering education: Views of U.S. faculty, chairs, and deans. *Journal of Engineering Education, 103*(2), 193-219.

[3] Rugarcia, A., Felder, R., Woods, D., & Stice, J. (2000). The future of engineering education. Part 1. A vision for a new century. *Chemical Engineering Education, 34*(1), 16-26.

[4] Felder, R. (2006). *A whole new mind for a flat world. Chemical Engineering Education, 40*(2). 96-97.

[5] Hart Research Associates (2013). It takes more than a major: Employer priorities for college learning and student success. An online survey among employers conducted on behalf of the Association of American Colleges and Universities.

[6] Archer, W., & Davison, J. (2008). Graduate employability. *The council for industry and Higher Education*.

[7] Berdrow, I., & Evers, F. T. (2010). Bases of competence: an instrument for self and institutional assessment. *Assessment & Evaluation in Higher Education*, *35*(4), 419-434.

[8] Watson, W. R., & Watson, S. L. (2013). Exploding the ivory tower: Systemic change for higher education. *TechTrends*, *57*(5), 42-46.

[9] ACM/IEEE-CS Joint Task Force on Computing Curricula 2013. Computer Science Curricula 2013. ACM Press and IEEE Computer Society Press.

[10] ABET. (2014). *Criteria for Accrediting Engineering Programs*. Retrieved from http://www.abet.org/Linked Documents-UPDATE/Criteria and PP/C001 08-09 CAC Criteria 11-8-07.pdf

[11] Caskurlu, S., Exter, M., & Ashby, I. (2016). *Importance of lifelong learning skills and degree covered in undergraduate programs: Faculty and practitioners' perspective*. Paper presented at the 2016 annual meeting of the American Educational Research Association, Washington, DC. Retrieved from the AERA Online Paper Repository.

[12] Exter, M. & Caskurlu, S. (N.d.). *Comparing job needs and undergraduate experiences*. Manuscript submitted for publication

[13] Exter, M. E. (2011). *The educational experiences of software designers working in education/instructional technology related fields* (Unpublished doctoral dissertation). Indiana University, IN.

[14] Exter, M. (2014). Comparing educational experiences and on-the-job needs of educational software designers. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education* (pp. 355-360). ACM.

[15] Exter, M., & Turnage, N. (2012). Exploring experienced professionals' reflections on computing education. *ACM Transactions on Computing Education (TOCE)*, *12*(3), 12.

[16] Lethbridge, T. C. (2000). What knowledge is important to a software professional? *Computer*, *33*(5), 44-50.

[17] Andriole, S. J. and Roberts, E. (2008). *Technology curriculum for the early 21st century*. Retrieved from http://cacm.acm.org/magazines/2008/7/5359-point-counterpoint-technology-curriculum-for-the-early-21st-century/fulltext

[18] National Academy of Engineering (NAE). (2004). *The engineer of 2020: Visions of engineering in the new century*. Washington, DC: National Academy of Engineering. Available from http://www.nap.edu/openbook.php?record_id=10999&page=10

[19] Kuh, G. D. (2008). High-impact educational practices: What they are, who has access to them, and why they matter. Report from the Association of American Colleges and Universities.

[20] Lester, S. (1999). An introduction to phenomenological research. *Stan Lester Developments*, 1-4.

[21] Groenewald, T. (2004). A phenomenological research design illustrated. *International Journal of Qualitative Methods, 3(*1). Article 4. Retrieved from http://www.ualberta.ca/~iiqm/backissues/3_1/pdf/groenewald.pdf

[22] Jones, S., Torres, V., & Arminio, J. (2014). *Negotiating the complexities of qualitative research in higher education* (2nd ed.). New York, NY: Routledge

[23] Patton, M. (2004). *Qualitative research and evaluation methods* (4th ed.). Thousand Oaks, CA: Sage.

[24] Kruger, D. (1988). *An introduction to phenomenological psychology* (2nd ed.). Cape Town, South Africa: Juta.

[25] Saldaña, J. (2015). *The coding manual for qualitative researchers*, Los Angeles, CA: Sage.

Appendix A.

1. Role

1. Where do you currently work?

    1.1. What is your concentration?

    1.2. How long have been working at XXXX University?

    1.3. Which courses do you teach?

    1.4. What is your official title at XXXX University?

    1.5. Could you describe your job as faculty member at XXXX University?

    1.6. What is your work history prior to starting this position?

        I directly came to graduate school right after undergraduate.

    1.7. So you don't have any industry experience?

2. Alignment

    2.1. To what degree do you think the formal education in CS programs prepare students to the job?

    2.2. Which courses are extensively taught? How are they important for the professional life?

    2.3. What were the most valuable things about the college/university classes stressed to prepare students to the job?

        2.3.1. How much do you stress these skills in your CS courses?

        2.3.2. How much are these skills stressed in the CS curriculum?

    2.4. What are the most important things you believe could have been improved about the college/university classes?

    2.5. What are the most important skills required on the job market?

        2.5.1. How much do you stress these skills in your CS courses?

2.5.2. How much are these skills stressed in the XXXX CS curriculum?

2.6. If you could design an ideal undergraduate [program type matching the one they teach in, e.g. computer science, software engineering, etc.] program, what would it be like?