AC 2011-862: SOFTWARE ENGINEERING, COMPUTER ENGINEERING, COMPUTER SCIENCE SIBLING DISCIPLINES WITH DIVERSE CULTURES

Susan E. Conry, Clarkson University

Susan E. Conry is Distinguished Service Professor of Electrical and Computer Engineering and Director of Software Engineering at Clarkson University. She is also currently the Chair Elect of the EAC of ABET. She received her Ph.D. from Rice University in Electrical Engineering.

Dr. Conry's research and educational interests cover various areas of computer engineering and software engineering. Her work in multiagent systems has focused on agent negotiation strategies, distributed constraint satisfaction problems, distributed genetic algorithms, and distributed search. Her work related to software engineering is concerned with patterns in hardware/software codesign. Dr. Conry has been designated Fellow of the IEEE, CSAB Fellow, and ABET Fellow. She was also recipient of the 2005 EAB Award for Meritorious Achievement Award in Accreditation Activities.

Software Engineering, Computer Engineering, Computer Science Sibling Disciplines with Diverse Cultures

Abstract

Every program of study has an institutional context. This institutional context influences its content, its relationships with other programs, and the character of the graduates, among other things. A recent study of the curricula of the 19 software engineering program that were accredited in October 2009 indicated that more than half of these programs were housed in the same department as an accredited computer science program. It also indicated that there is a wide degree of variation in the degree of overlap in content of the software engineering curriculum and that of the computer science curriculum. At some institutions, there was as little as one course overlap between the two programs while at others there was a very large component that was shared. At some of these institutions, computer science and software engineering shared treatment of introductory material while others offered even introductory material to computer science students and software engineering students in different course sequences. This evidence suggests that there may be well be interesting dimensions of commonality and difference among curricula in a broader range of areas that were treated by the *Computing Curricula 2005* documents.

This paper extends the previous study to the consideration of interrelationships among the curricula of computing programs at a subset of institutions with accredited software engineering programs in October 2010. It considers the computing disciplines more broadly, including software engineering, computer science, and computer engineering programs, as these computing disciplines may be regarded as having stronger interrelationships due to greater commonality in technical topical content. The subset treated includes only those institutions in the United States with accredited software engineering programs at which there are also programs in the other two "major computing disciplines" and does not address the content of other computing disciplines such as information technology, information systems, or management information systems. There are thirteen such institutions. Of these institutions, the majority (eight) involve institutional contexts in which all of the responsible departments are in the same school or college. Within this set of eight institutions, some lodge all three of the disciplines (computer science, software engineering, and computer engineering) in the same department while others house them in different departments. At three of the institutions, computer science is lodged in a school that has no other engineering programs. The remaining two institutions either do not have an administrative structure equivalent to a school or a college or have separate colleges of computing and engineering.

Interrelationships in at least three dimensions are discussed in this paper: relationships evidenced by administrative housing within the institution, relationships involving common sets of required courses, and relationships exposed by varying types and degrees of overlap among the programs. Those areas that are distinctive about a given program on a given campus will also be discussed, as they provide just as much insight into the relationships among the programs as areas of commonality do.

Introduction

Today, computing touches the life of every person on the planet. Practitioners in the computing disciplines are constantly devising systems that have impact on our financial systems, our economy, our understanding of medical issues, progress in discovery of new scientific principles, the way we communicate with one another and the way we relax. The people who work in the computing disciplines have been educated in a variety of disciplinary contexts, three of which are relevant to this study: computer engineering, computer science, and software engineering. There is still come controversy about the relationship among these computing disciplines. This paper is concerned with exploring the relationships between them by focusing on the curricular structure and content of baccalaureate programs in the disciplines that have some commonality in institutional context. Historically, the computer engineering, computer science, and software engineering disciplines emerged at different times and were nourished by different parent disciplines[1].

Prior to about 1970, someone who wanted to enter the computing profession could study in one of three areas: computer science (if they wanted to focus on software), electrical engineering (if hardware was of interest to them), or information systems (if their focus was on business applications). With the development of microprocessors in the 1960's, the character of the landscape began to change. Technology changed, creating demand for engineers who understood the hardware and electronics underlying the chips but also were conversant with and capable of developing the software components of a system. It was not possible to adequately treat the topics needed for education of these engineers in the context of a specialization area within an electrical engineering program of study. The first computer engineering program was accredited by the EAC of ABET in 1971, and between about 1970 and 1990, computer engineering emerged as a separate discipline. The last decade of the twentieth century saw computer engineering firmly established as an independent discipline.

Computer science programs were in their infancy in the early 1970's, and there was significant controversy in those early years as to whether or not computer science really was a legitimate academic discipline. Some saw it as an arm of applied mathematics, others saw it as an attempt to provide some venue for educating programmers, and still others saw it as a true academic discipline. Programs in computer science had wildly divergent structure and content. Some were lodged within departments of mathematics and others were not. It was possible for a student to get a degree in computer science in a program that included only coursework involving programming in two or three languages. As the discipline matured, a measure of commonality was fostered by the efforts of the ACM in devising curriculum guidelines and the development of accreditation criteria by guided by the ACM and the IEEE Computer Society through CSAC. The first computer science programs were accredited by CSAC of CSAB in 1986. By the 1990's, there was no longer controversy concerning legitimacy of the discipline.

The term "software engineering" was coined in 1968, and there were isolated courses in software engineering offered on some campuses as early as the mid-1970's, but the first software engineering programs developed in the United States were graduate programs whose students were expected to already have acquired substantial expertise in computer science. Gradually, as the importance of software engineering topics was recognized, treatment of software engineering concepts filtered into undergraduate programs. Sometimes these concepts were woven into

courses in computer engineering and sometimes they were treated in computer science courses. As more experience was gained with software engineering courses within computer science curricula and computer engineering curricula, it was recognized that it took a different experience to educate a software engineer[1]. Education of a software engineer requires coursework and applied project experience that goes beyond what can be added to a computer science curriculum and it takes inclusion of a larger set of computer science topics than can be added to a computer engineering curriculum. Thus software engineering emerged as an independent discipline. The first baccalaureate programs in software engineering in the United States were developed in the mid to late 1990's, and programs in software engineering have been accredited by the EAC of ABET since 2002.

Clearly, computer engineering, computer science, and software engineering, are closely related disciplines with overlapping content areas. The historical context tells us something about the relationships of these disciplines when taken from a developmental perspective, but the relationships among the disciplines as they exist on university campuses in the United States today are not clear. In the sections that follow, we provide evidence concerning these relationships. We consider those institutional contexts within which a software engineering program accredited by the EAC of ABET is offered and in which there is also a computer science program and a computer engineering program. Our focus of attention is on the curricular context and content and administrative structure within which these programs exist. Two dimensions are of particular significance as we consider the interrelationships between programs on a given campus: administrative structure within which the programs are housed and the curricula – commonality and lack thereof in the programs of study. In the sections that follow, we discuss each of the dimensions and the ways in which they reveal how the programs are intertwined and the ways in which they are distinct.

Administrative Structure

The character of any program of study is strongly influenced by the background and philosophy that is brought to the program by its faculty, and that can often be characterized by the administrative structure within which the program is housed. In general, programs that are housed in schools or colleges that also house other engineering programs may be more strongly influenced by an engineering perspective than those that are housed in other units. The programs at the thirteen institutions considered in this study are housed in widely divergent administrative units. Some of them reside in a school of arts and sciences with other departments in mathematics and the physical sciences. Others are in units that also house a wide spectrum of engineering programs. Some are the responsibility of departments that are responsible for a single academic program and others are lodged in the same department as the other two computing programs considered. Table 1 provides information concerning the relevant degree program names, the accredited status of the programs, the housing department (if applicable) and the housing school or college within the institution. As is clear from Table 1, at eight of these institutions (just over 60%) all three programs are resident within a school or college of engineering. At three of the institutions (about 23%) of the institutions, computer science is lodged in a school or college with other science departments. In one institution there is a college of computing that houses computer science and software engineering while the school of engineering is home to the computer engineering program and one institution has no schools. It

Institution	Degree Program Name	Accredited?	Housing Department Name	School or College Unit
Cal Poly, San Luis Obispo	BS in Computer Science	CAC	Computer Science and Software Engineering	College of Engineering
	BS in Computer Engineering	EAC	Interdisciplinary, Computer Science and Electrical Engineering Departments	College of Engineering
	BS in Software Engineering	EAC	Computer Science and Software Engineering	College of Engineering
Clarkson	BS in Computer Science	No	Mathematics and Computer Science	School of Arts and Sciences
	BS in Computer Engineering	EAC	Electrical and Computer Engineering	School of Engineering
	BS in Software Engineering	EAC	Shared between Electrical and Computer Engineering and Computer Science	School of Engineering
Drexel	BS in Computer Science	CAC	Computer Science	College of Engineering
	BS in Computer Engineering	EAC	Electrical and Computer Engineering	College of Engineering
	BS in Software Engineering	EAC	Shared between Computer Science and Electrical and Computer Engineering	College of Engineering

Table 1. Administrative Characteristics

Embry Riddle	BS in Computer Science	No	Electrical, Computer and Software Engineering	College of Engineering
	BS in Computer Engineering	EAC	Electrical, Computer, and Software Engineering	College of Engineering
	BS in Software Engineering	EAC	Electrical, Computer and Software Engineering	College of Engineering
Fairfield	BS in Computer Science	no	doesn't appear to be a department	College of Arts and Sciences
	BS in Computer Engineering	EAC	doesn't appear to be a department	School of Engineering
	BS in Software Engineering	EAC	doesn't appear to be a department	School of Engineering
Florida Institute of Technology	BS in Computer Science	CAC	Computer Sciences	College of Engineering
	BS in Computer Engineering	EAC	Electrical and Computer Engineering and Computer Sciences	College of Engineering
	BS in Software Engineering	EAC	Computer Sciences	College of Engineering
Michigan Dearborn	BS in Computer and Information Science	CAC	Computer and Information Science	College of Engineering and CS
	BS in Computer Engineering	EAC	Electrical and Computer Engineering	College of Engineering and CS
	BS in Software Engineering	EAC	Computer and Information Science	College of Engineering and CS

Table 1. Administrative Characteristics (cont.)

Mississippi State	BS in Computer Science	CAC	Computer Science and Engineering	School of Engineering
	BS in Computer Engineering	EAC	Electrical and Computer Engineering and Computer Science and Engineering	School of Engineering
	BS in Software Engineering	EAC	Computer Science and Engineering	School of Engineering
Penn State Erie, Behrend College	BS in Computer Science	no	No distinct departments identified	School of Science
	BS in Computer Engineering	EAC	No distinct departments identified	School of Engineering
	BS in Software Engineering	EAC	No distinct departments identified	School of Engineering
RIT	BS in Computer Science	CAC	Computer Science	College of Computing and Information Science
	BS in Computer Engineering	EAC	Computer Engineering	College of Engineering
	BS in Software Engineering	EAC	Software Engineering	College of Computing and Information Science

Table 1. Administrative Characteristics (cont.)

Rose Hulman	BS in Computer Science	CAC	Computer Science and Software Engineering	no schools
	BS in Computer Science	EAC	Computer Engineering	no schools
	BS in Software Engineering	EAC	Computer Science and Software Engineering	no schools
UT Arlington	BS in Computer Science	CAC	Computer Science and Engineering	College of Engineering
	BS in Computer Engineering	EAC	Computer Science and Engineering	College of Engineering
	BS in Software Engineering	EAC	Computer Science and Engineering	College of Engineering
UT Dallas	BS in Computer Science	CAC	Computer Science	Engineering and Computer Science
	BS in Computer Engineering	EAC	Computer Engineering	Engineering and Computer Science
	BS in Software Engineering	EAC	Software Engineering	Engineering and Computer Science

Table 1. Administrative Characteristics (cont.)

is interesting to note that only one of the computer science programs in institutions housing that program in a school of engineering is not accredited by the CAC of ABET while all of the programs considered in this study that are lodged in schools of science or arts and science are not accredited by the CAC of ABET. Based on this data, it would appear that accreditation of computer science programs is not viewed as being important in those institutions where computer science is lodged outside the school of engineering.

Curricular Characteristics

A number of questions arise when one considers curricula. How much of the curriculum is devoted to technical topics? Of the technical component, how much is required and how much is elective? Are these two characteristics related to the discipline? Are they related to the housing administrative unit? The answers to these questions can provide some indication as to the ways in which the intrinsic character of the disciplines are similar. To the extent that there is agreement among the disciplines in a particular dimension, independent of institutional positioning factors, one could say that the factor is indicative of disciplinary similarities. To the extent that these factors seem to be correlated with administrative housing, perhaps the institutional context has shaped the character of the program more.

Table 2 summarizes the data relative to the overall content of the technical curricular component and the degree to which it reflects required coursework. Table 2(a) gives the relevant data for computer engineering programs, while Table 2(b) and Table 2(c) summarize this information for computer science and software engineering programs. Some interesting patterns emerge when this data is analyzed. First, the relative size of the technical component in the computer engineering and software engineering programs is similar – an average of about 51% of the total curriculum is technical in nature. The variation in the size of the technical component is also not great – the maxima do not vary much and neither do the minima. The difference between the maximum and minimum sizes of the technical component is also similar – around 15 percentage points. The computer science programs are much more widely divergent. The average technical component for computer science programs is smaller than the average for the engineering programs considered by about 5 percentage points, but there is a much wider swing between the maximum and the minimum – on the order of 46 percentage points. In addition, it is evident in the data that the four computer science programs that are not lodged in the same academic unit as engineering have the smallest technical components by a significant margin. This suggests that the general shape of the curriculum – its allocation of time and attention to technical concepts supporting the discipline – is influenced by where it is lodged in the administrative structure of the institution. In general, faculties in schools of arts and sciences are more likely to develop programs of study that allow students significantly more elective choices. Engineering programs, on the other hand, are generally more focused and allow far fewer elective choices.

Relationships Among Curricula

- **Required Coursework in Computing Shared by All Disciplines**

Given the observation that the overall curricular content structure appears to be influenced to some extent by the administrative structure within which the program is housed, the next question that one might ask concerns the degree to which the required portion of the technical

CpE Institution	Tech Component as % of Total	Req Tech as % of Tech	Tech Elect as % of Tech	Housing Unit
1	49.2%	89.8%	10.2%	Engineering
2	60.9%	84.6%	15.4%	Engineering
3	54.3%	82.6%	17.4%	Engineering
4	45.5%	90.2%	9.8%	Engineering
5	52.3%	91.3%	8.7%	Engineering
6	52.8%	78.8%	21.2%	Engineering
7	54.7%	91.4%	8.6%	Engineering
8	48.8%	85.7%	14.3%	Engineering
9	53.5%	92.5%	7.5%	Engineering
10	52.6%	80.4%	19.6%	No Schools
11	48.5%	87.4%	12.6%	Engineering
12	50.4%	90.6%	9.4%	Engineering
13	48.4%	100.0%	0.0%	Engineering
Average	51.7%	88.1%	11.9%	
Max	60.9%	100.0%	21.2%	
Min	45.5%	78.8%	0.0%	

Table 2(a) Computer Engineering Overall Curricular Content

CS Institution	Tech Component as % of Total	Req Tech as % of Tech	Tech Elect as % of Tech	Housing Unit
4	31.7%	60.5%	39.5%	A&S
1	35.8%	72.1%	27.9%	A&S
9	37.9%	66.7%	33.3%	Computing
8	39.3%	81.3%	18.8%	Science
5	40.2%	88.2%	11.8%	Engineering
7	43.8%	78.6%	21.4%	Engineering
2	46.1%	72.1%	27.9%	Engineering
12	46.3%	78.6%	21.4%	Engineering
11	49.2%	73.0%	27.0%	Engineering
13	49.6%	85.5%	14.5%	Engineering
10	50.0%	75.0%	25.0%	No Schools
6	56.7%	58.8%	41.2%	Engineering
3	77.7%	29.8%	70.2%	Engineering
Average	46.5%	70.8%	29.2%	
Max	77.7%	88.2%	70.2%	
Min	31.7%	29.8%	11.8%	

Table 2(b) Computer Science Overall Curricular Content

SwE Institution	Tech Component as % of Total	Req Tech as % of Tech	Tech Elect as % of Tech	Housing Unit
1	49.2%	89.8%	10.2%	Engineering
2	46.8%	79.5%	20.5%	Engineering
3	56.7%	83.3%	16.7%	Engineering
4	43.9%	89.7%	10.3%	Engineering
5	47.2%	90.0%	10.0%	Engineering
6	53.3%	78.1%	21.9%	Engineering
7	53.9%	82.6%	17.4%	Engineering
8	52.4%	86.4%	13.6%	Engineering
9	53.3%	65.4%	34.6%	Computing
10	50.0%	83.3%	16.7%	No Schools
11	46.4%	73.0%	27.0%	Engineering
12	49.2%	89.8%	10.2%	Engineering
13	58.9%	71.2%	28.8%	Engineering
Average	50.9%	81.7%	18.3%	
Max	58.9%	90.0%	34.6%	
Min	43.9%	65.4%	10.0%	

Table 2(c) Software Engineering Overall Curricular Content

component is shared and to what extent the required technical courses differ from one of these disciplines to another. The data regarding the degree of overlap between required technical courses in the computer engineering, computer science, and software engineering programs reveals some interesting characteristics. Figure 1 shows the degree of commonality among the programs at the 13 institutions covered in this study. The comparative data make it clear that there is a greater fraction of the required technical component that computer science programs share with the other two disciplines on almost all of the campuses involved in the study. There is one campus on which there are no shared required courses and there are two on which more than 70% of the required technical courses are common courses required of students in all three of these disciplines. The average shared technical content is 24.5% of the total technical component. There is no clear correlation between the position in administrative structure and the degree of commonality of technical content. The computer science programs in institutions 1, 4, 8, and 9 are not housed in the same administrative unit as any engineering programs, yet the degree of commonality among programs in those institutions is not markedly different from the others – they are not outliers as far as shared content is concerned.

As Figure 1 clearly shows, the software engineering and computer engineering programs track each other fairly closely in the degree of commonality among the programs while the computer science shared content is, in general, somewhat higher. The aggregate data reveals that on average, computer engineering programs have 24.5% shared technical content with the other two

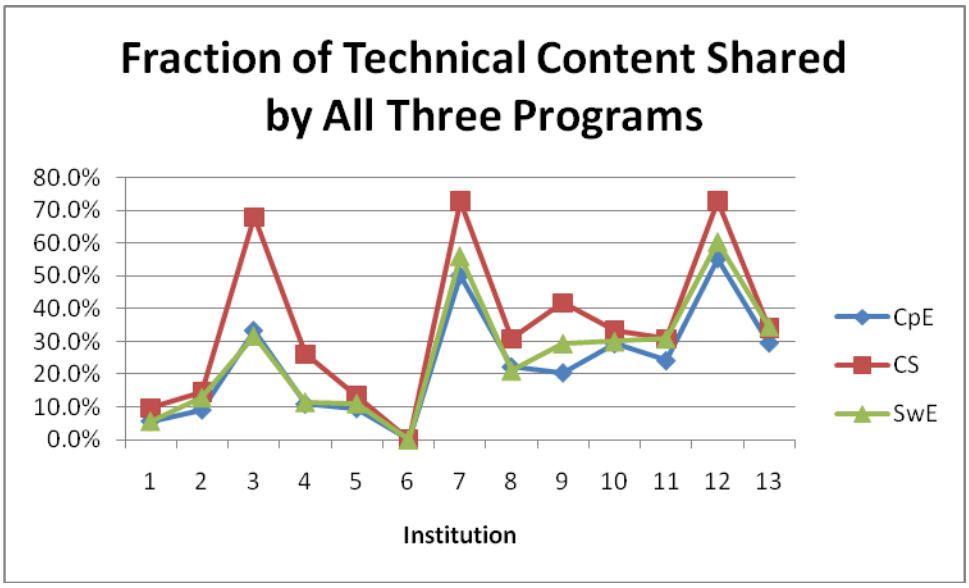


Figure 1: Fraction of Required Technical Content Shared by all Three Programs

disciplines while computer science programs have 36.5% and software engineering programs have 27.5%

Why should this be the case? One answer could be found in the set of names of courses that are found in the set of common courses. That set of course names is found in Table 3. Clearly, most of the commonality is found in the mathematical foundations of the disciplines and in the general area of data structures and algorithms. About 40% of the programs in this study require students in all three programs to take the same courses in operating systems and software engineering. One feature of the data in this table is striking. Only two of these institutions require students in all three programs to take a common course named Computer Science 1. Based on their course descriptions, one could argue that courses named "Introduction to Computer Programming" or "Introduction to Programming Techniques" or "Problem Based Introduction to Computer Science" are logical equivalents to a course named "Computer Science 1". Considering all of these courses as "Computer Science 1" raises the number of programs sharing treatment of elementary programming concepts, but the number is still less than 40% of the programs in the study. Clearly, students graduating with degrees in any of these disciplines must be able to write computer programs. There must be something about disciplinary differences or cultural differences between academic units on different campuses that are driving development of courses that provide different exposure to these concepts to students in different computing majors.

- **Required Technical Component Shared by Two of the Three Disciplines**

One can gain some insight into the degree of disciplinary interaction by considering the extent to which the program of study in each of these disciplines shares content with the other two beyond what they all have in common. Thus, for example, if we found that the computer science programs shared a large fraction of their non-common required technical content with computer

Number of Institutions	Title of Shared Course
11	Discrete Math, Discrete Structures, or Applied Algebra
7	Data Structures or Data Structures and Algorithms
6	Operating Systems
5	Software Engineering
3	Computer Science 2
3	Digital Circuits or Logic Design
3	Intermediate Programming
3	Object Oriented Programming
2	Computer Organization or Computer Architecture
2	Computer Science 1
1	Advanced Programming Tools and Techniques
1	Computer Science 3
1	Computer System Design Project I
1	Computer System Design Project II
1	Distributed Client Server Programming
1	Introduction to Engineering
1	Introduction to Analysis of Algorithms
1	Introduction to Computer Programming
1	Introduction to Computing for Engineers
1	Introduction to Computer Science & Engineering
1	Introduction to Programming Techniques
1	Introduction to Software Development
1	Microprocessors
1	Problem Based Intro to CS
1	Professional Practices

Table 3: Names of Required Courses Shared by All Three Disciplines

engineering programs, we could infer that the two disciplines are closely linked with one another. If we observed that they have little required additional technical content in common we would be inclined to conclude that they are not. Figure 2 shows the fraction of the technical content of each of the programs and its pairwise comparison with the two others. This data does not include those courses that are in common with all three disciplines but is reflective of what ***else*** is shared between them. A careful reader will notice that the degree of overlap depicted in this figure is not symmetric. Thus, for example, the fraction of overlap between software engineering and computer science reported for institution 6 in Figure 2(b) is not the same as the overlap reported between these disciplines in Figure 2(c). This is due to the fact that the size of the required technical components of these two programs is not the same. In general, computer

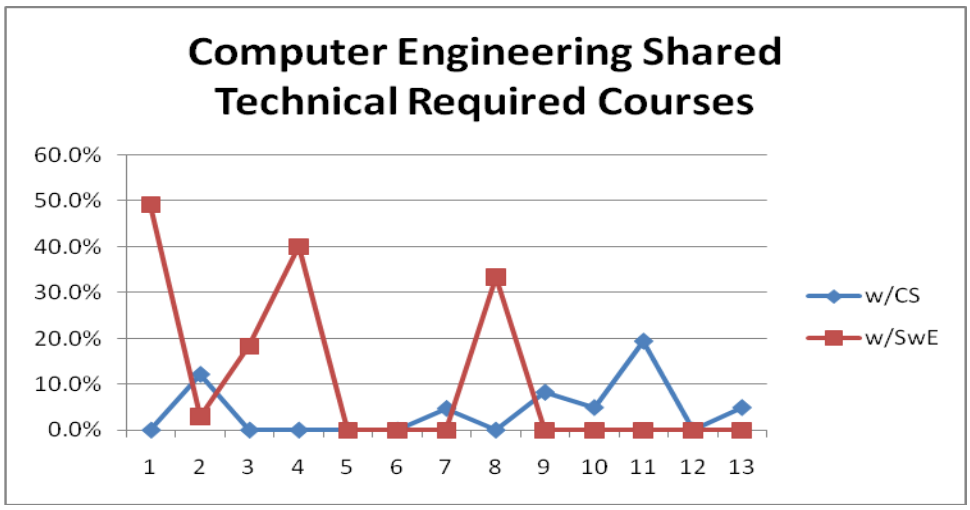


Figure 2(a)

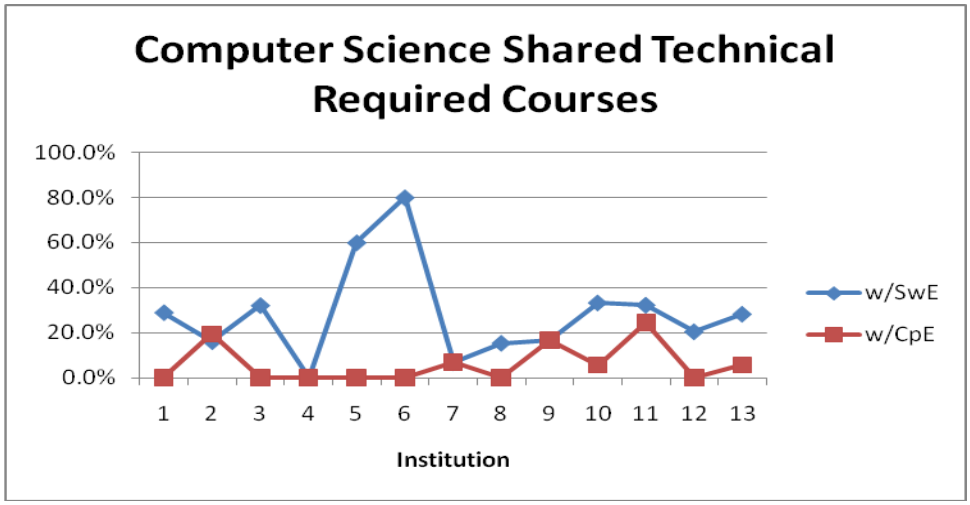


Figure 2(b)

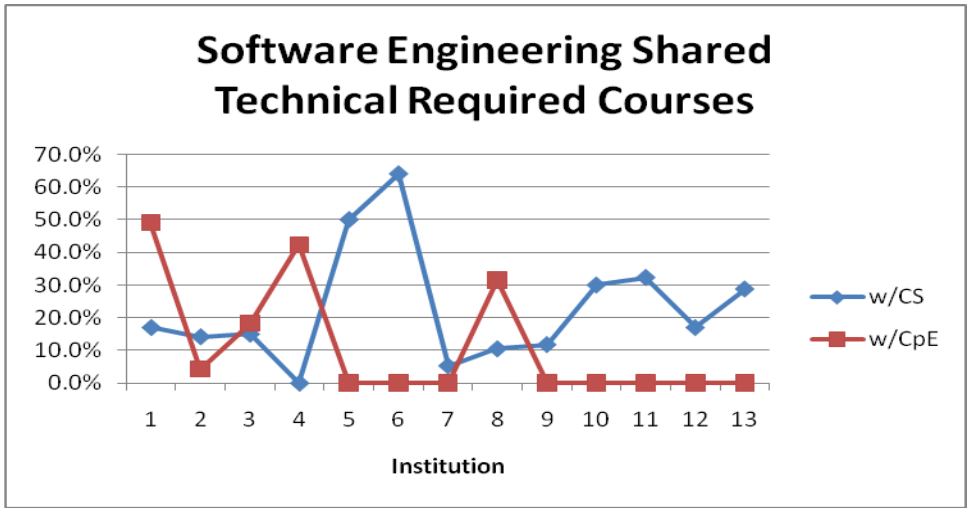


Figure 2(c)

science programs have a somewhat larger set of elective choices in the technical component than do the engineering programs.

As is evident from Figure 2(a), there is very little additional required technical content that is shared only between computer engineering and software engineering on most of these campuses. There is also little shared additional required content shared only between computer engineering and computer science programs. Taken pairwise, the largest degree of overlap, as reflected in the fraction of the required technical component that is shared by only those two disciplines is found between computer science and software engineering. The aggregate data is shown in Table 4. Clearly on some campuses there is a very high degree of interrelationship between computer science and software engineering but this is far from a universal phenomenon.

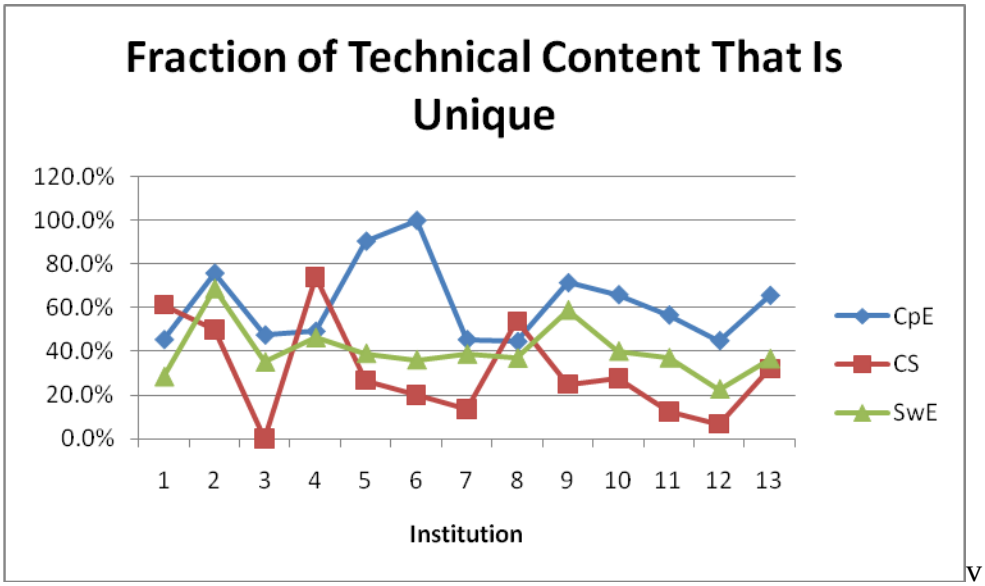
	SwE-CS	SwE-CpE	CS-CpE	CS-SwE	CpE-CS	CpE-SwE
Avg	23.2%	8.0%	6.6%	28.5%	4.5%	7.9%
Max	60.4%	49.1%	24.6%	80.0%	19.3%	49.1%
Min	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%

Table 4: Aggregate Data - Pairwise Comparison of Fraction of Required Technical Component Shared Between Majors

- **Required Technical Content that is Unique**

We have considered that portion of the required technical content of the curriculum that is shared by students in all three of the computing majors on a given campus and those portions that are shared by two of the three. The data concerning these factors give us insight into the inter-relationships between and among these three computing disciplines. But each of these computing disciplines has characteristics that distinguish it from the other two. How is this reflected in the required component of the curricula in question?

Figure 3 shows the fraction of the required technical component that is unique to the program in question. So, for example, for institution 1, 61.3% of the required technical component in computer science is unique to the computer science program – not required for either computer engineering or software engineering. The overall impression conveyed by this figure is that the computer engineering programs in general have a larger fraction of their technical content that is unique. This is not surprising – there is almost always a significant component of a computer engineering program that treats topics in digital and analog electronics and other areas of electrical engineering that are seldom treated in computer science or software engineering programs. The software engineering programs have a required technical component that is more often unique than those of the computer science programs on their campuses. This, too, is to be expected. Because the computer science programs generally have a larger elective component, it is frequently the case that these electives encourage students to explore areas of the discipline of interest to them, while the software engineering students are focused more specifically on those topics of relevance to the education of a software engineer.



v

Figure 3: Fraction of the Required Technical Component Unique to Each Program

Table 5 shows the average, maximum, and minimum fraction of the required technical content of the curricula that are unique to the discipline.

	CpE	CS	SwE
Avg	63.1%	28.5%	41.2%
Max	100.0%	73.9%	68.6%
Min	44.4%	0.0%	22.6%

Table 5: Aggregate Data – Fraction of Required Technical Component Unique to Discipline

Just as the names of the courses that were required of students in all three majors on a given campus provides us with insight concerning commonality, the names of courses that are unique can tell us about the unique character of each discipline. There are over 275 different names for courses that are unique to these programs on different campuses, so there is little to be gained by listing them here. For the purposes of considering the character of the disciplines involved, course names that convey similar content are considered "one".

Most of the computer engineering programs require courses in digital and analog circuits, logic design and digital systems design, computer architecture, microcontrollers or microprocessors, signals and systems, and embedded systems. All of the computer engineering programs include engineering design sequences in their required component. Few of the programs in computer science or software engineering require courses in these areas, though several of the computer science curricula do require a computer organization course – it just is not the same course as the one required for the computer engineers.

Topical areas that appear to be required by a number of computer science programs but not computer engineering or software engineering in any significant number include automata

theory, theory of computation, compiler construction, and programming languages. A number of computer science programs do require a course in software engineering and courses in computer organization and digital logic – just not the same courses that are required of the students in the relevant other major.

Software engineering students are frequently required to take courses in database systems, networks, requirements engineering, project management, software architecture, software testing, verification, and validation, and software quality assurance. As is the case with the computer engineering programs, there is always a software engineering design sequence in the required portion of the curriculum.

Concluding Remarks

This paper has explored the interrelationships between and among programs in computer engineering, computer science, and software engineering as they are reflected in their administrative housing within the institution and the nature of the curriculum. The curricula in computer engineering, computer science, and software engineering were characterized along a number of dimensions. The first set of dimensions was concerned with the size of the technical component of the entire program of study and, within the technical component how much of that content is required and how much elective. The issue of whether this curricular "shape" appeared to be a function of discipline and/or whether it was influenced by the administrative housing of the program was also discussed.

Similarities and differences in disciplines are often reflected in the relationships between and among the required components of the programs of study. In this paper we discussed the size and the content of the set of courses that is required by all three of the disciplines on a given campus. This allowed us to infer something about the character of the set of topics that are shared by all of these disciplines. It also allowed us to observe that even though a set of concepts may be common to all of the disciplines, there may well be differences in the treatment of these concepts from one discipline to another. This is reflected in the relative lack of commonality in treatment of elementary programming concepts. Going beyond the set of common required courses, one can gain some insight by considering the sets of courses that are required by two of the three computing disciplines. This reveals relationships reflected in overlap between two of the disciplines that are not common to the third. This data revealed that there is generally more commonality between computer science and software engineering curricula than there is between either of these two disciplines and computer engineering. It also revealed that there is, in an aggregate average sense, only about 25% of the technical program of study beyond the common required courses that is shared between the computer science and software engineering programs. The disciplines clearly have components of their curricula that distinguish them from one another.

The component of the technical curricula that most clearly distinguishes the character of these computing programs lies in the required component that is unique to each of the programs. The size of the "unique" technical component and its content clearly reflect the differences. When viewed from the perspective of the size of the unique component, it becomes clear that the discipline with the largest such component is computer engineering, followed by software engineering then computer science. Though the relative magnitude of the size of this component varies from campus to campus, the computer engineering program invariably has the largest

unique component. On all but three of the campuses, the software engineering program has a larger unique component than the computer science program. Perhaps it is just a coincidence, but the three programs in which the computer science component has a larger unique component are those which are not resident in a school or college that houses any engineering programs.

What does this tell us about the disciplines – sibling disciplines with different cultures? It tells us that there is demonstrably a body of skills and knowledge that is required of students in all three disciplines, but students in each of these disciplines may acquire many of these skills and much of this knowledge in courses that are not necessarily taken by students in the other two disciplines. It tells us that there is a commonality between the computer engineering and software engineering programs that is reflected in the general shape of the curriculum, in the degree to which student elective choices are constrained, and in the emphasis on engineering design. These are programs that are intended to prepare students for professional careers. It tells us that there is in general more commonality between the computer science and software engineering programs than there is between either of these two programs and the computer engineering program on the same campus. This study also shows us something about the unique character of each of the disciplines. The sets of topics treated in required courses that are unique to each discipline show us that the computer engineer is more focused on topics integrating hardware and software in computing systems, that the software engineer is more focused on software process, software design principles and practices, and software quality issues, and that the computer scientist must learn topics concerning theoretical concepts, programming languages, and compilers that are not required of the others. Sibling disciplines, indeed, but with differing underlying cultures and distinctive attributes.

Bibliography

1. Computing Curricula 2005: The Overview Report http://www.acm.org/education/curric_vols/CC2005-March06Final.pdf accessed 18 January, 2011.
2. Accredited Programs, http://www.abet.org/AccredProgramSearch/AccreditationSearch.aspx , accessed 18 January, 2011.
3. Source of the curricular data and course names and content was obtained from university catalogs available at:
    - http://www.calpoly.edu/
    - http://www.clarkson.edu/
    - http://www.drexel.edu/
    - http://www.erau.edu/
    - http://www.fairfield.edu/
    - http://www.fit.edu/
    - http://www.msstate.edu/
    - http://psbehrend.psu.edu/
    - http://www.rit.edu/
    - http://www.rose-hulman.edu/
    - http://www.umd.umich.edu/
    - http://www.uta.edu/uta/
    - http://www.utdallas.edu/