6-1-2012

Software Engineering Education Needs More Engineering

A. Frank Ackerman, Ph.D.
Montana Tech

Sushil Acharua, D.Eng.

AC 2012-3372: SOFTWARE ENGINEERING EDUCATION NEEDS MORE ENGINEERING

Prof. A. Frank Ackerman, Montana Tech of the University of Montana

A. Frank Ackerman has 50 years of experience in all phases of software development. In 1985, he founded the Institute For Zero Defect Software to do applied research, consulting, and training for software development organizations seeking to improve the reliability of their software. His personal experience has lead him to the conviction that today's development organizations can achieve significant improvement in software reliability for a small increase in effort. Some of his current research and educational activities are focused on improving current specification, coding, test, and review techniques for the development of high quality software. Ackerman has been active in either the ACM or the IEEE throughout his career. He is a Life Member of the IEEE. Presently, he is an Associate Professor of software engineering at Montana Tech of the University of Montana. He is a graduate of the University of Chicago and holds a Ph.D. in computer science from the University of North Carolina, Chapel Hill.

Dr. Sushil Acharya, Robert Morris University

Sushil Acharya, D.Eng., Associate Professor of software engineering, joined Robert Morris University in the spring of 2005 after serving 15 years in the Software Industry. With U.S. Airways, Acharya was responsible for creating a data warehouse and using advance data mining tools for performance improvement. With i2 Technologies, he worked on i2's Data Mining product "Knowledge Discover Framework" and at CEERD (Thailand), he was the Product Manager of three energy software products (MEDEE-S/ENV, EFOM/ENV and DBA-VOID), which are in use in 26 Asian and seven European countries by both governmental and non-governmental organizations. Acharya has a M.Eng. in computer technology and a D.Eng. in computer science and information management with a concentration in knowledge discovery, both from the Asian Institute of Technology in Thailand. His teaching involvement and research interests are in the areas of software engineering and development (verification and validation) and enterprise resource planning. He also has interest in learning objectives-based education material design and development. Acharya is a co-author of "Discrete Mathematics Applications for Information Systems Professionals," 2nd Ed., Prentice Hall. He is a life member of Nepal Engineering Association and is also a member of ASEE and ACM. Acharya is a recipient of the "Mahendra Vidya Bhusak" a prestigious medal awarded by the Government of Nepal for academic excellence. He is a member of the Program Committee of WMSCI, MEI, CCCT, EEET, ISAS, AG, KGMC, and IMCIC and is also a member of the Editorial Advisory Board of the Journal of Systemics, Cybernetics, and Informatics of the International Institute of Informatics and Systemics.

Software Engineering Education Needs More Engineering

Abstract
To what extent is "software engineering" really "engineering" as this term is commonly understood? A hallmark of the products of the traditional engineering disciplines is trustworthiness based on dependability. But in his keynote presentation at ICSE 2006 Barry Boehm pointed out that individuals', systems', and peoples' dependency on software is becoming increasingly critical, yet that dependability is generally not the top priority for software intensive system producers. Continuing in an uncharacteristic pessimistic vein, Professor Boehm said that this situation will likely continue until a major software-induced system catastrophe *similar in impact to the 9/11 World Trade Center catastrophe* stimulates action toward establishing accountability for software dependability. He predicts that it is highly likely that such a software-induced catastrophe will occur between now and 2025.

It is widely understood that software, i.e., computer programs, are intrinsically different from traditionally engineered products, but in one aspect they are identical: the extent to which the well-being of individuals, organizations, and society in general increasingly depend on software. As wardens of the future through our mentoring of the next generation of software developers, we believe that it is our responsibility to at least address Professor Boehm's predicted catastrophe.

Traditional engineering has, and continually addresses its social responsibility through the evolution of the education, practice, and professional certification/licensing of professional engineers. To be included in the fraternity of professional engineers, software engineering must do the same. To get a rough idea of where software engineering currently stands on some of these issues we conducted two surveys. Our main survey was sent to software engineering academics in the U.S., Canada, and Australia. Among other items it sought detail information on their software engineering programs. Our auxiliary survey was sent to U.S. engineering institutions to get some idea about how software engineering programs compared with those in established engineering disciplines of Civil, Electrical, and Mechanical Engineering. Summaries of our findings can be found in the last two sections of our paper.

1. Introduction

The debate over whether or not "software engineering" is a legitimate branch of engineering has been going on since the term first appeared in the professional literature in1968 [1,2]. Naturally, as with any new concept, the debate begins with confusion over just what the term refers to. For starters, "software" has several meanings [3]. In this paper we use this term to refer to a computer program product. Such a product must include a file, or files, that execute on a stored programmed computing machine. Typically such a product also includes (1) text file(s) of source code from which the executable file(s) are automatically constructed, and (2) a variety of different kinds of "documents" that describe, among other things, how the executing program should perform, how the source code is structured, the non-functional requirements of the executable and source files, the processes and techniques used to create the product, and a variety of user information, including how to report any problems that are experienced during program execution. Under this definition a software product is essentially the same as any other

engineered product. One major difference is that ultimately software is nothing but enormous streams of individual computer instructions that each execute in pico-seconds, so unlike all other engineered products, a software product does not have any actual physical existence. [note 1] So to claim that such a product is "engineered" stretches the accepted definition of "engineering" [4] to cover new territory. Another important difference is that unlike any other engineered product, a software product is not manufactured, at least not in the sense that physical products are. A software product is easily, and for negligible cost, flawlessly duplicated as many times as desired. This attribute, however, only underscores the importance of the product's engineering, for essentially it is just the engineering that produces the product.

This whole discussion would just be an academic exercise except for the fact that more and more the correct functioning of practically every other socially important engineered product depends on the correct functioning of at least one, and sometimes hundreds of computer programs. This problem was first recognized in military systems, and so the term "software engineering" was created to address it. Since 1968 the "software problem" has intensified many times over to the point where software is now an Achilles heel of our entire civilization, as the quote in our abstract from the eminent Professor Boehm points out.

In the past the driving force behind the formalization of the traditional engineering disciplines was similar to the current effort to formalize software engineering, as without the formalization of the various traditional engineering disciplines buildings and bridges often collapsed, mines caved in, chemical plants blew up, etc. [5]. [note 9] Yet forty-three years after the term "software engineering" was first coined, the debate over whether or not software engineering is a legitimate branch of engineering rages in some academic and professional circles. Meanwhile, millions of people world-wide are daily subjected to the damages caused by poorly engineered software [6].

In 1993 Fletcher Buckley laid out for the IEEE Board of Governors what was required to transform the then current, mostly haphazard, production of software products into a responsible branch of engineering. [note 12] In a follow-up piece in *Computer* [7] Mr. Buckley enunciated three major objectives that needed to be met:

1. the establishment of software engineering as an approved [academic] program, included the associated accreditation issues;
2. the establishment of a separate set of software engineering ethics; and
3. the establishment of software engineering as a certified or registered field of engineering.

Not mentioned in his *Computer* piece, but also discussed by the IEEE Board was the need for

4. the creation of a comprehensive set of widely accepted Software Engineering standards.

All of these objectives except the establishment of certification or registration have now been achieved. This has been achieved in only a few jurisdictions [note 2], but in 2009 the National Council of Examiners for Engineering and Surveying approved an effort to develop a Professional Engineer examination for software engineers that is now well underway [8]. According to the current schedule, it is anticipated that the first licensing examination for software engineering will be available for administration sometime in 2013 [9]. As this effort moves forward, however, some of the weaknesses in the accreditation criteria for academic software engineering programs will begin to surface.

One of the foundation elements of every profession is a consensus among its members about what constitutes the core body of knowledge and skills that make up the basic competency of members of that profession. Over the years a lot of effort has been put into defining this core competency for software engineering. There are two major results of this effort: (1) the *Software Engineering 2004: Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering* [10], and (2) the *Guide to the Software Engineering Body of Knowledge, 2004 Version* [27]. The curriculum guidelines describe the entire body of knowledge that should be covered in an undergraduate program in software engineering. This knowledge is referred to as the SEEK (Software Engineering Educational Knowledge). On the other hand, the guide document is intended to cover just the software engineering knowledge that a software engineer should have after four years of professional practice, and is referred to as the SWEBOK (Software Engineering Body of Knowledge). As the subject of this paper is undergraduate software engineering education, we leave any discussion about the SWEBOK and the relationship between the SEEK and the SWEBOK for another venue.

The objectives of this paper are:

- to get some idea of the extent to which *Software Engineering 2004* [note 4] is used as the basis for undergraduate degree programs in software engineering,
- to get some feeling for the extent to which the academic software engineering community thinks that *Software Engineering 2004* is an adequate basis for undergraduate degree programs in software engineering, and
- to try and determine the extent to which current undergraduate degree programs in software engineering "line up" with similar programs in engineering generally.

To address these objectives, in addition to reviewing the literature on the professionalization of software engineering, we constructed a survey to gather information on what academic software engineering programs around the world were offering to undergraduates. We used SurveyMonkey [11] as the host for our survey and emailed requests to take our on-line survey to more than a hundred academics world-wide. Thirty-one academics replied to our survey. A detailed description of what these thirty-one people reported to us is given in the survey section below. We then attempted to relate the software engineering material our respondents reported as currently being taught to undergraduates to the material that is covered in traditional undergraduate engineering programs. Our conclusions and recommendations for further efforts in providing future software engineering graduates with the knowledge and skill they need to create trustworthy software products conclude this paper.

Currently many more institutions offer undergraduate degrees in Computer Science than offer degrees in Software Engineering, and due to the current shortage of recipients of Software Engineering degrees, society's need for software is often served, albeit haphazardly, by recipients of Computer Science degrees. So before we jump into the analysis of our survey results, in the next section we describe why we believe that a degree in Computer Science does not provide an adequate foundation for a career as a professional software engineer.

2. Computer Science versus Software Engineering

Wikipedia defines "computer science" as "the study of the theoretical foundations of information and computation. It also includes practical techniques for their implementation and application in computer systems" [12]. Wikipedia defines "software engineering" as "the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software, and the study of these approaches; that is, the application of engineering to software" [13]. From these two definitions it can be deduced that (1) computer science is one of the foundation science disciplines for software engineering, as physics is for electrical engineering [14]; (2) there is significant overlap between the two disciplines in that both include practical techniques for the implementation and application of computer systems; and (3) that one of the disciplines focuses on science, and the other on engineering. In as much as academic programs in Computer Science significantly predate programs in Software Engineering, differentiating Software Engineering from Computer Science, and especially the establishment of separate undergraduate degrees in software engineering has been a contentious issue in academia.

Several prominent software engineers and writers on software engineering have addressed the difference between computer science and software engineering. Over many years David Parnas has been a powerful and vociferous advocate for software engineering as a much different discipline than computer science: *The important issue is the content and style of the education. University programs in engineering are very different from programs in the sciences, mathematics, and liberal arts. These disparities derive from the differences in the career goals and interests of the students. ... Future scientists must learn:*

- *what is true (an organized body of knowledge about the phenomena of interest)*
- *how to confirm or refute models of the world relate to that growing body of knowledge, and*
- *how to extend the knowledge of what is true in their field.*

In other words, scientists learn science plus the scientific methods needed to extend science. On the other hand, future engineers, who will be responsible for designing trustworthy products, must learn:

- *what is true and useful in their chosen specialty (an organized body of knowledge),*
- *how to apply that body of knowledge,*
- *how to apply a broader area of knowledge necessary to build complete products that must function in a real environment, and*
- *the design and analysis disciplines that must be followed to fulfill the responsibilities incumbent upon those who build products for others.*

In other words, engineers learn science plus the methods needed to apply science [14].

Steve McConnell sums it up this way: *Universities award computer science degrees, and they normally expect their computer science students to start building real-world software. This puts the students in a technological no-man's land. They are called scientists, but they are performing job functions that are traditionally performed by engineers without the benefit of engineering training* [15].

3. Main Survey Results and Commentary

3.1. Survey Introduction

To understand the breadth and depth of current Software Engineering Bachelor degree programs a survey was conducted using SurveyMonkey [11]. The text of our survey can be viewed http://cs.mtech.edu/main/images/surveys/se2011survey.pdf It contains 268 questions but not all questions are presented to every respondent. A respondent that was not at an institution that offered a bachelors program in software engineering could complete the survey by answering less than 21 questions. Email requesting participation in our survey was sent to 102 individuals at institutions that we had reason to believe offered software engineering courses. In a few instances we sent our message to more than one individual at an institution. Filling out our survey was often more than a matter of just a few minutes, so we believe that it is safe to assume that we received only one response from an institution unless the responses were for different programs. Altogether we emailed to 90 institutions. In our email message we requested recipients to forward our message to any other colleagues that might be helpful in assessing the state of practice in undergraduate Software Engineering education, so our message may have gone to recipients that were not on our email list.

We gave our respondents the opportunity to identify themselves and or their institution but only 5 (other than ourselves) identified themselves or their institution.

We sent email to institutions listed in Box 1. From the 102 emails we sent out, 31 responded to the survey. Since we wanted to get as much information as we could, our survey was structured so that respondents who did not have time to provide all the information we requested could still provide some information and complete the survey.

A person taking any survey can of course stop answering questions at any point. SurveyMonkey keeps track of all respondents who started the survey and also those that completed the survey by clicking on the survey's final closing button. In our case only 16 of the respondents that started the survey completed it.

3.2. Overview of Survey Responses

The first few questions of the survey provided background information on the participating institutions.

- 100% of those who responded) said their institution offer courses in Software Engineering. From this group 42% (13) offer a B.S. degree in Software Engineering, 13% (4) offer a B.S. degree in Engineering with Software Concentration, 32% (10) did not offer a B.S. degree with Software Engineering in the title, and the remaining 13% (4) offered a variety of options.

- Of the 13 respondents that offer a B. S. in Software Engineering 46% (6) offer this degree through their Computer Science Department. 39% (5) offer this degree through an engineering department, and 15% (2) offer this degree through departments other than computer science or engineering.

Box 1: List of Institutions

- Adelaide University
- Auburn University (2 messages)
- Cal Poly
- Capitol College
- Carleton University
- Carnegie Mellon University
- Central Connecticut State University
- Champlain College
- Clarkson University
- Columbus State University
- Concordia University
- CS, University of Victoria
- Dept. of CS University of Alberta
- Dept. of CS University of Calgary
- Drexel University
- Electrical and Computer Engineering at the University of Alberta
- Embry-Riddle Aeronautical University
- Fairfield University
- Florida Institute of Technology
- Gannon University
- Grand Valley Stage University
- Harvard University
- Iowa State University
- Lakehead University
- Lyndon State College
- Marshall University
- McGill University
- McMaster University (2 messages)
- Milwaukee School of Engineering
- Milwaukee School of Engineering
- Mississippi State University
- Mississippi State University
- Monash University
- Monmouth University
- Montana State University
- Montana Tech
- Navel Postgraduate School
- Norfolk State University
- North Carolina A&T State University
- Oklahoma City University
- Penn State Erie
- Penn State University
- Princeton University
- Princeton University
- Princeton University
- Purdue University
- Robert Morris University
- Rochester Institute of Technology
- Rochester Institute of Technology
- Rose-Hullman
- Salt Lake Community College
- San Jose State University
- Santa Clara University
- Software Engineering, University of Victoria
- South Dakota State University
- Southern Polytechnic State University
- St. Mary's University
- St. Edward's University
- Stanford University
- Stevens Institute
- SUNY Plattsburg
- The University of Tennessee
- The University of Texas at Arlington
- The University of Texas at Dallas
- The University of Texas at El Paso
- UC Berkeley
- University of Calgary (2 messages)
- University of Florida
- University of Houston Clear Lake
- University of Maryland (2 messages)
- University of Melbourne
- University of Michigan - Dearborn
- University of New Brunswick
- University of New South Wales (2 messages)
- University of North Carolina (2 messages)
- University of North Florida (2 messages)
- University of Ontario Institute of Technology
- University of Oregon
- University of Ottawa
- University of Pittsburgh
- University of Regina
- University of Washington (3 messages)
- University of Waterloo
- University of Western Australia
- University of Western Ontario
- University of Wisconsin at Platteville (2 messages)
- University of Wollongong (3 messages)
- USC
- UVA Wise
- Virginia State University

- Of the 13 respondents that offer a B. S. in Software Engineering 85% (11) are ABET (or equivalent) accredited. [note 5]

- Of the 5 respondents who reported offering a B. S. in Engineering with a concentration in Software Engineering, to the question: "*Is your Bachelors Degree in Engineering with a concentration in Software Engineering accredited by ABET or an equivalent organization?"* [note 8]; 40% (2) said they were accredited and 60% (3) said that they were not.

– There were 18 responses to the question: "Does your institution also offer a Masters degree in Software Engineering?" The responses were exactly evenly split.

3.3. Software Engineering Courses for Other Degree Programs

Respondents who said that their institutions did not offer a bachelors degree with "Software Engineering" in the title, or who chose "Other" were asked questions about which courses they offered and whether or not these courses were offered by the Computer Science department, or engineering department, or other department. The table below summarizes the courses and the department that offers them. To make it easy for a respondent to complete our survey we provided the courses listed below and instructed the respondents to select a course if they had a course with that title or thought they had a similar course with different title. We also provided the opportunity for respondents to list alternative or additional courses. Only one respondent made any additional entries (3) and he/she did not enter any course titles. We believe these entries can be ignored.

For each of the courses a respondent selected he/she had to choose one of the options (1) offered by the Computer Science department, (2) offered by an engineering department, or (3) offered by another department. Table 1 shows the totals for each choice for each course title that we listed in the survey.

This was the only information we collected from respondents that indicated they did not offer a Bachelors degree in Software Engineering or a Bachelors degree in Engineering with a concentration in Software Engineering. After providing this information these respondents were branched to the survey's concluding questions.

Table 1: Software Engineering Courses for Other Degree Programs

Course	CS Department	Engineering Department	Other Department
Introduction to Software Engineering and Computing	2	1	
Software Engineering and Computing II	2		1
Software Engineering and Computing III	1		1
Introduction to Software Engineering	5	1	
Software Construction	1		1
Software Engineering Approach to Human Computer Interaction	1		2
Software Design and Architecture	3		1
Software Quality Assurance and Testing	2		1
Software Requirements Analysis	1		2
Software Project Management		1	2
Design and Architecture of Large Software Systems	1		1
Software Testing	3		1
Low Level Design Software	2		1
Software Process Management		1	2
Formal Methods in Software Engineering			1
Software Engineering Capstone Project	3		2

3.4. Coverage of SEEK Knowledge Areas [note 10]

For respondents who indicated that they did offer a Bachelors degree in Software Engineering, or a Bachelors degree in Engineering with a concentration in Software Engineering, we asked respondents to identify the SEEK areas covered by each of the SEEK recommended Software

Engineering courses for the Bachelors degree in Software Engineering[10]. We instructed our respondents to select a SEEK course title if they thought one of their course with a different title was similar. For each course, along with the choices for selecting SEEK area coverage we provided the option "Course did not touch any SEEK area". In hindsight we can see that we should have included some additional options for each course. Had we done so we might have collected more information about offered courses even though our respondents could not relate them to SEEK categories.

The responses were not as numerous as the authors had hoped. Table 2 shows the number of responses for each recommended course. On average only 5 respondents assigned SEEK areas. In our email message (sent by a third party to preserve the authors' anonymity) we told our recipients we would be asking them to relate their courses to the SE2004 report and in the survey itself we provided links to summary information about the SEEK areas. For those recipients that were familiar with SEEK the information we asked for would not have been difficult to provide. Those recipients not familiar with SEEK or how their courses related to the SE2004 guidelines quite possibly felt that for the purpose at hand it was not worth their time to become familiar with the SE2004 guidelines. [note 6].

Table 2: Coverage of SEEK Knowledge Areas – Responses

Course	Responses Received
1. Introduction to Software Engineering and Computing	6
2. Software Engineering and Computing II	5
3. Software Engineering and Computing III	5
4. Introduction to Software Engineering	7
5. Software Construction	4
6. Software Engineering Approach to Human Computer Interaction	4
7. Software Design and Architecture	8
8. Software Quality Assurance and Testing	6
9. Software Requirements Analysis	4
10. Software Project Management	4
11. Design and Architecture of Large Software Systems	3
12. Software Testing	3
13. Low Level Design Software	3
14. Software Process Management	3
15. Formal Methods in Software Engineering	3
16. Software Engineering Capstone Project	6
17. Data Structures and Algorithm	7
18. Programming Fundamentals	6
19. Object Oriented Paradigm	6
20. Discrete Structures	6
21. Statistics and Empirical Methods	6

In Table 3 we take a look at the data that was supplied by the few recipients that answered this group of questions. The column headings are the SEEK Knowledge Areas and the row headings

are the courses. SEEK Knowledge Areas covered in a course identified by 50% or more of the respondents is represented by a X. The rightmost column lists the number of respondents who felt the course did not touch any SEEK Knowledge Area.

3.4.1. Summary - Coverage of SEEK Knowledge Areas

Table 4 depicts the cumulative responses by course by SEEK knowledge areas. Course numbers 1 to 21 corresponds to the courses listed in Table 2.

Chart 1 depicts Table 4 in a bar chart. The response from the survey shows that all SEEK Knowledge Areas are covered by the 21 software engineering courses listed in Table 2. However the survey does not indicate that the software engineering courses delivered at an institute covers all SEEK Knowledge Areas. Based on the responses Software Design has been selected to be covered the most in the software engineering courses followed by Software Modeling & Analysis and Software Quality. However Software Evolution and Software Practice have been selected to be covered the least in the software engineering courses. The chart also indicates that a total of 5 respondents could not map their course to a SEEK Knowledge Area.

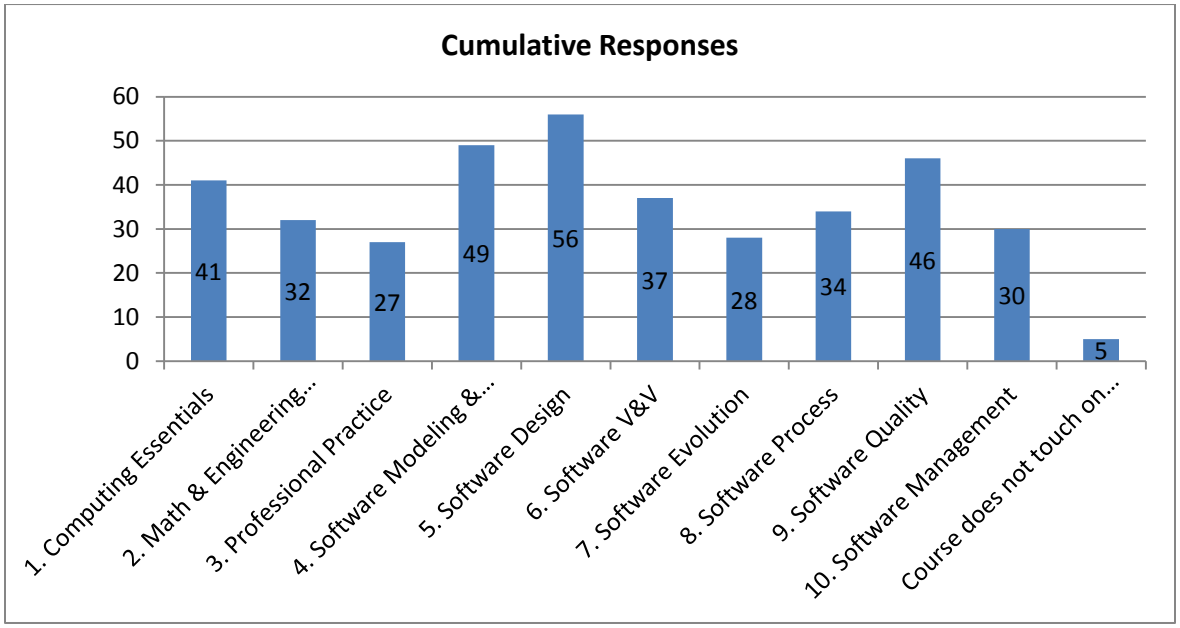
Table 3: SEEK Knowledge Areas Covered in a Course Identified by 50% or More Respondents

SEEK Knowledge Areas →	1. Computing Essentials	2. Mathematical and Engineering Fundamentals	3. Professional Practice	4. Software Modeling and Analysis	5. Software Design	6. Software Verification and Validation	7. Software Evolution	8. Software Process	9. Software Quality	10. Software Management		# of respondents mentioning - course does not touch any SEEK area
Course ↓	8	3	6	12	11	6	5	7	10	6	**← No. of Responses ↓**	
1. Introduction to Software Engineering and Computing	X			X	X						6	
2. Software Engineering and Computing II	X			X	X			X			5	
3. Software Engineering and Computing III				X		X					5	
4. Introduction to Software Engineering			X	X	X	X		X	X	X	7	1
5. Software Construction			X		X	X	X		X		4	
6. Software Engineering Approach to Human Computer Interaction			X	X	X				X		4	1
7. Software Design and Architecture				X	X				X		8	1
8. Software Quality Assurance and Testing						X	X		X	X	6	
9. Software Requirements Analysis			X	X			X	X		X	4	
10. Software Project Management			X					X		X	4	
11. Design and Architecture of Large Software Systems				X	X				X		3	
12. Software Testing						X	X	X	X		3	
13. Low Level Design Software	X			X	X						3	1
14. Software Process Management								X	X	X	3	
15. Formal Methods in Software Engineering		X		X					X		3	
16. Software Engineering Capstone Project	X	X	X	X	X	X	X	X	X	X	6	
17. Data Structures and Algorithm	X										7	
18. Programming Fundamentals	X				X						6	
19. Object Oriented Paradigm	X			X	X						6	
20. Discrete Structures		X									6	
21. Statistics and Empirical Methods	X										6	1

Table 4: Cumulative Responses by Course by SEEK Knowledge Areas

	# of	Courses																				
	Responses	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
1. Computing Essentials	41	5	3	2	2	1	1	0	0	1	0	0	1	2	0	1	3	6	6	4	2	1
2. Math & Engineering Fundamentals	32	1	2	0	2	1	0	1	0	1	0	1	1	1	0	2	3	3	1	1	6	5
3. Professional Practice	27	1	1	1	4	2	2	0	1	2	2	1	1	1	1	1	5	0	1	0	0	0
4. Software Modeling & Analysis	49	3	4	4	6	1	2	5	1	4	1	2	1	2	0	3	3	1	1	3	1	1
5. Software Design	56	3	3	2	6	4	3	7	0	1	1	3	1	2	0	1	5	3	4	5	1	1
6. Software V&V	37	1	1	4	5	2	1	3	6	1	1	1	3	0	1	1	5	0	1	0	0	0
7. Software Evolution	28	1	1	2	2	2	0	3	3	2	1	1	2	0	1	1	4	0	1	1	0	0
8. Software Process	34	1	3	2	6	0	0	2	2	2	3	0	2	0	3	1	4	0	2	1	0	0
9. Software Quality	46	2	2	2	5	2	2	4	6	1	2	2	2	1	3	2	6	0	1	1	0	0
10. Software Management	30	1	1	2	4	0	0	0	4	2	4	0	1	0	2	1	6	0	1	1	0	0
Course does not touch on any SEEK areas	5	0	0	0	1	0	1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	1

Chart 1: Cumulative Responses by Course by SEEK Knowledge Areas



3.5. Other Software Engineering Courses Offered

As we did for the software engineering courses offered by non-SE degree institutions, respondents from degree granting institutions were given the opportunity to list alternative or additional courses required for a Bachelors degree in software engineering. Six responses were received for this option. These additional required courses and the SEEK Knowledge Area they cover are listed in Table 5.

Chart 2 depicts the number of times each of the SEEK areas was cited in the required software engineering courses that the respondents listed.

3.6. SEEK Coverage of Knowledge Areas for a Bachelors Degree in Software Engineering

After obtaining information about required software engineering courses and their relationship to the SEEK areas we asked "In your opinion, to what extent does SEEK cover all the knowledge areas necessary for a Bachelors education for competent, socially responsible engineers?" We offered a choice of one of the following:

- Has no serious deficiencies
- Has some serious deficiencies
- Has many serious deficiencies
- Don't know enough about SEEK
- No opinion

Table 5: Additional Courses and SEEK Knowledge Areas

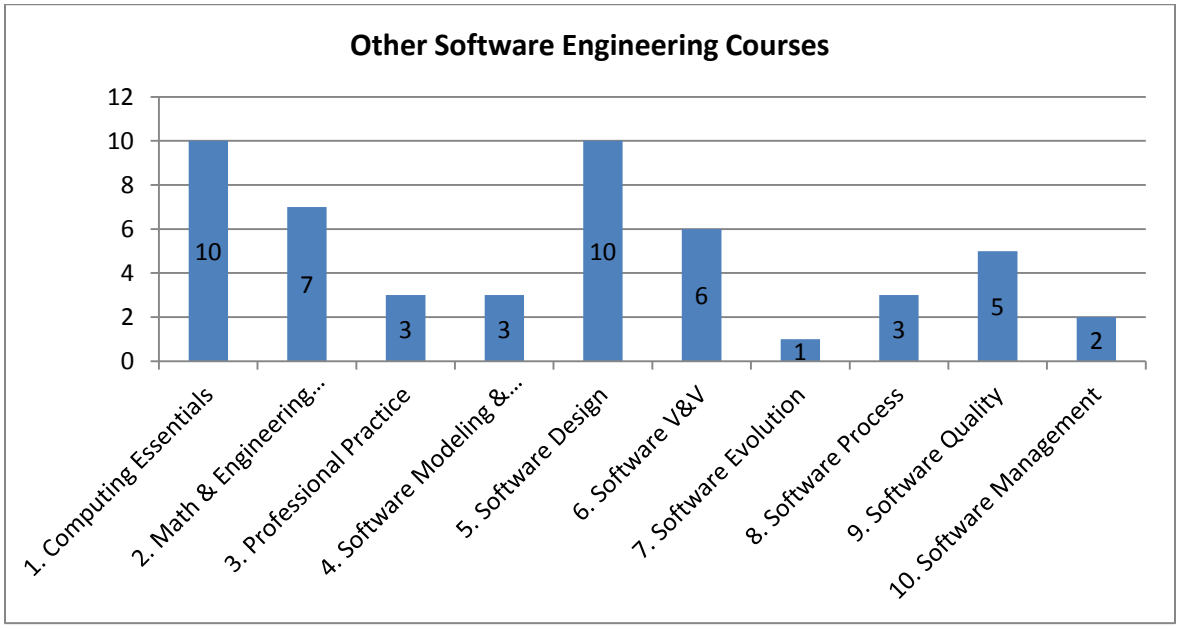
SEEK Knowledge Areas →	1. Computing Essentials	2. Mathematical and Engineering Fundamentals	3. Professional Practice	4. Software Modeling and Analysis	5. Software Design	6. Software Verification and Validation	7. Software Evolution	8. Software Process	9. Software Quality	10. Software Management
Course ↓										
Computer Architecture for Software Engineers		x			x					
Concepts of Programming Languages		x								
Database Design	x									
Design and Analysis of Algorithm	x									
Design of Operating Systems	x				x					
Discrete Structures		x								
Engineering Economics Analysis		x								
Fundamentals of Computer Science II	x				x	x			x	
Intro to Embedded Systems	x									
Introduction to Computer Networking	x									
Introduction to Software Process								x	x	
Introduction to Software Verification					x	x				
Introduction to Team Software Development								x	x	
Operating Systems	x				x					x
Presenting Technical Information			x							
Real Time Systems	x	x		x	x	x				
Real-time Embedded, Software Maintenance and Reengineering		x		x	x					
Software Component Design					x					
Software Engineering Economics	x	x	x	x	x	x	x	x	x	x
Software Maintenance			x			x			x	
User Interface Design					x	x				
Web Application Design and Implementation	x		x		x					
Web Science	x									

Five of the respondents found no deficiencies with the SEEK coverage. Four respondents indicated that the current coverage areas have serious deficiencies. If the respondent indicated that he/she felt that seek had deficiencies we asked for further clarification. The comments we received are listed in Box 2.

Box 2: SEEK Coverage Deficiencies

- Heavy on management. Not enough on different kinds of systems, e.g.: operating systems, web applications. We are focused on satisfying the ABET criteria. These criteria are too general to target specific SEEK topics.
- SEEK is rather outdated and the discipline of SE has advanced significantly since it was created. On the flip side, nice to have something stay constant, but that is very rare in the fast paced world of computing.
- No clear distinction of B.S. in Software Engineering and B.S. in Engineering - Software Engineering Track.
- The Issue of Professional Engineering (Software Engineering Licensing) is not touched. Maybe in 2004 this was a non-issue.

Chart 2: Other Software Engineering Courses



3.7. Software Engineering Electives

After asking about required courses and their coverage of the SEEK areas we asked "Can a student in your Software Engineering program take any Computer Science or Software Engineering courses other than the required courses you selected or listed above?" Nine respondents answered this question. One responded that in his/her program there were no electives. For those that responded that their program included electives we gave them the choice of (1) providing a URL to information about these electives, (2) emailing our "alter ego" information about their electives, or (3) listing their electives. Five responded by providing a URL, two responded by listing their electives on the survey, and one promised to email our "alter ego" but did not. Box 3 depicts a composite list of the electives that students in software engineering programs may take in the 7 programs that responded.

Box 3: Composite List of Electives

- Advanced Networking
- Algorithms
- Algorithms Analys/Design
- Algorithms and Data Structures II
- Analysis and Design of Computer Communications Networks
- Analysis of Algorithms
- Artificial Intelligence (3)
- Cloud Computing
- Component Based Software Engineering
- Computational Biology Algorithms
- Computer and Network Security (2)
- Computer Animation
- Computer Ethics
- Computer Game Design I
- Computer Game Design II
- Computer Graphics (3)
- Computer Organization
- Computer Supported Collaborative Work
- Computer Systems and Architecture
- Compiler Construction
- Compiler Design
- Concurrency
- Cryptography
- Data Base Systems (2)
- Data Mining
- Database Management Systems
- Decision Sup & Expert Sys
- Device Control
- Digital Forensics
- Digital Forensics II
- Digital Signal Processing
- Digital Signal Processing: II
- Distributed Computing
- Distributed Systems and the Internet
- Embedded Systems
- Fault-Tolerant Computing
- Global, Economic, Society, Ethical Issues in Computing
- Healthcare Information Systems
- Human Factors in Engineering
- Image Processing
- Industrial Robots
- Information Security
- Information Systems
- Information Systems Desg
- Introduction to Computer Gaming
- Introduction to Computer Graphics
- Introduction to C#
- Introduction to Game Design
- Introduction to Parallel and Cluster Computing
- Introduction to Geographical Information Systems
- Linux System Administration
- Management of Software Development
- Managing Software Development
- Media Applications
- Microelectromechanical Systems
- Microprocessors
- Mobile Application Development
- Modernizing Legacy Software
- Multimedia Systems
- Multiresolution Signal and Geometry Processing
- Network-centric Computing
- Network Security (2)
- Networks/Data/Computer Communications
- Numerical Methods (2)
- Object-Oriented Design
- Object-Oriented Programming
- Open Source E-com Development (LAMP)
- Operating Systems Concepts
- Operations Research: Linear Programming
- Operations Research: Simulation
- Organization of Prog Lang
- Overlay and Peer-to-Peer Networking
- Practice of Information Security
- Programming Languages (2)
- Robotics
- Robotics and Automation
- Secure Software Methods
- Simulation
- Software Architecture
- Software for Embedded and Mechatronics Systems
- Software Processes
- Sustainable Energy systems Design Project
- Switching, Network Traffic and Quality of Service
- System-on-Chip Engineering for Signal Processing
- System Reliability
- Testing and Quality Assurance
- Theory of Operating Systems
- Web-Based Client/Server Prog
- Web Technology
- Wireless and Mobil Networks

3.8. Communications, Math, Science and General Engineering Courses

After gathering information on required and elective courses for software engineering programs we asked: "Are your software engineering students required to take any Communications, Math, Science and General Engineering Courses?". Box 4 lists these courses.

Box 4: Communication, Science, Mathematics and General Engineering Courses

Communications

- Advanced Technical Writing
- Arguments and Research
- Business Professional Communications
- College Writing
- Communications (2)
- Design and Communication
- Freshman Composition I
- Freshman Composition II
- Intercultural Communications
- Presenting Technical Information
- Reading and Writing Strategies

Mathematics

- Calc I
- Calc II
- Calc III
- Differential Equations
- Matrix Algebra for Engineers
- Multivariate Calculus

Science

- Astronomy
- Biology
- Chemistry I and Lab
- Fundamentals of Chemistry
- Fundamentals of Physics
- Geology
- Physics I and Lab
- Physics II and Lab
- Physics III and Lab

General Engineering (Basic)

- Circuits and Electromagnetic
- Engineering Economic Analysis
- Engineering Fundamentals I
- Engineering Graphics
- Engineering Success Skills
- Introduction to Engineering (2)
- Mechanics for Engineers
- Operations & Product Management
- Static and Strength of Material

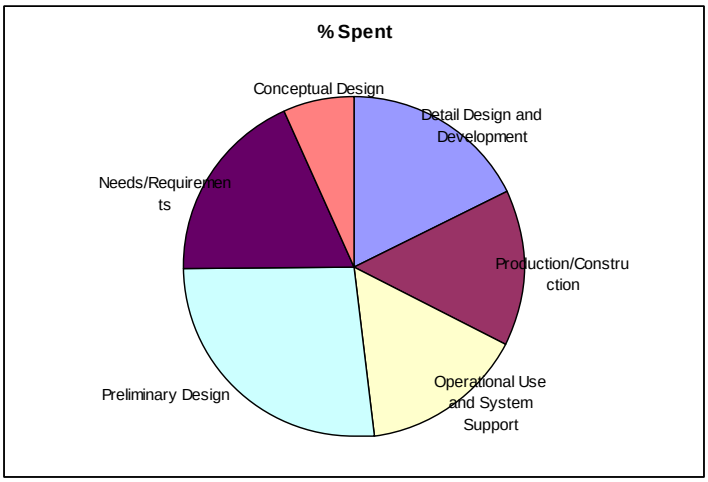
3.9. Systems Engineering

In our quest to gather information on how the courses required for software engineering degrees aligned with those for traditional engineering disciplines we asked respondents to estimate the percentage of time their software engineering students spent in each of the following System Engineering areas. We received 7 responses. The data in Table 6 is the average of these responses. Chart 3 is a pie chart representation of Table 6.

Table 6: Systems Engineering Coverage

Phase	% Spent
Needs/Requirements	18
Conceptual Design	15
Preliminary Design	15
Detail Design and Development	27
Production/Construction	18
Operational Use and System Support	7

Chart 3: % Time Spent on Systems Engineering Phases



3.10. Industrial Strength Software Development Environments in Education

We were curious about the extent to which bachelors level academic software engineer programs have been able to provide their students with "real" problems and development environments. We asked:

> *Educating traditional engineers often involves using scale model mock-ups of the artifacts, systems and environments they will encounter in practice. The analogue for software engineers might be industrial strength software development environments and teaching system/applications. In the box below please describe the software development environment and systems/applications your institution uses in its software engineering courses*

Box 5 lists the responses received.

Box 5: Responses to 3.10 Question

- The Capstone Project involves real customer and real problems.- Sharing of past student projects and their outcomes- Guest speakers.
- Case Studies in low level and medium level design – a tool used for Software Engineering Exercises.
- Engineering Projects in Community Service (EPICS) - a tool for students to complete SE projects for real clients.
- Windows, Linux, MS Visual Studio, MS Office (Word, PowerPoint, Project, Visio), Enterprise Architect, Bugzilla, Tortise SVN, Oracle
- Eclipse Ubuntu / gnu UNIX Beagleboard
- IBM/Rational tools, Subversion and Visual SourceSafe, Microsoft Visual Studio 2010, NetBeans

3.11. Licensing Software Engineers

One of the major differences between many of the traditional engineering disciplines and software engineers is that presently, at least in the U.S., there is only one state that has a licensing program for software engineers. [note 11] As we did further research for this paper we found that this difference is now being rigorously addressed [9]. We asked our respondents to address what we think is still the ACM's official opposition to licensure. [note 7] The responses listed in Box 6 are in line with current efforts to eliminate this difference between traditional engineering and software engineering.

Box 6: Responses to Software Engineering Licensing

- Considering (1) that software now controls almost every aspect of our industrial society, and (2) that all of the elements required for professional status are now in place (and have been so for more than five years) it is now time for software engineers to be licensed. Furthermore, state licensing exams would focus attention on the core information that we should be teaching.
- Only way SE will ever come to be considered a "true" engineering discipline.
- I am really neutral on it - it can be a good idea (but not if applicants will be taking current PE exam as part of their licensure)
- There are a lot of unqualified people doing software engineering and it makes it very hard to tell whether the software can be made to meet the needs of a customer. Software is involved in many mission critical aspects of our lives (avionics, automotive, medical, etc.) and we have little knowledge as to the skills and capabilities of the people writing this software.
- I feel it is time to revisit the policy. The society is more and more dependent on software and as engineers it is important to assure the society that the software they use is reliable, secure and functions the way it should. As engineers it is time for software engineers to be accountable and responsible, it is not that they are not, however a license could be a factor.

3.12. Survey Concluding Remarks

We concluded our survey by asking our respondents for any comments they would like to make about this survey, software engineering education in general, or Software Engineering as a recognized engineering discipline. The three remarks we received is listed in Box 7.

Box 7: Respondents' Comments

- I'm a bit concerned that the results of this survey will be very broad and inconclusive, especially given the questions up front. From the initial e-mail, I thought the survey was tailored to my institution, yet when I actually started answering the questions it became clear that this is not the case. Thus, the results may be very seriously flawed.
- It was very difficult mapping your course titles to the course titles used at our university. Because of that, I feel that some of my SEEK characteristic placement was not very accurate.
- Well rounded survey.

4. Comparison with Traditional Engineering Programs

To get some idea of how Software Engineering curriculum compared with major established engineering disciplines, namely Civil, Electrical, and Mechanical Engineering, a short ancillary survey was conducted. For each engineering discipline the ten top discipline specific institutions as categorized by US News [16, 17, 18] (category 1) and the ten top schools by undergraduate

student enrollment as categorized by US News [19] (category 2) were surveyed (a number of institutions fell into multiple categories). The former category was chosen since they provided quality engineers to the workforce and the latter was chosen since they provided quantity engineers to the workforce. The participants were surveyed on the engineering design content of their curriculum, FE examination preparation and participation, capstone projects, internships, and Systems Engineering. As all the institutes were ABET accredited they were not asked questions related to coverage in the sciences, mathematics, basic engineering and core engineering areas.

A total of forty auxiliary surveys were sent out to the department heads, directors, or deans of the institutions described above. Thirty percent of the institutions responded to the survey with most responses being received in the electrical engineering discipline from seven category 1 institutions. Below we summarize the responses relevant to the subject of this paper.

Engineering Design: All the institutes surveyed said design is an important component covered in multiple courses in their engineering curriculum. Some courses were more design heavy than others. 28% said less that 25% of the curriculum focused on design, 56% said between 25% and 50%, and 17% said more than 50% of the curriculum focused on design.

Internships: Most of the institutions surveyed said engineering internship is highly encouraged but 95% said internships are not part of their curriculum. 5% said internships are counted towards the degree.

Fundamentals of Engineering Exam: Most of the institutions surveyed said taking the FE exam is important and emphasized in the curriculum. 94% responded that the institute does not provide any support in terms of tutoring, and FE fees. These institutions indicated that students know that they need to appear for the FE exam and are mentally prepared for it. Only one institution surveyed mentioned that they provide tutoring services.

Capstone Projects: 100% of the institutes surveyed said capstone projects (either one or two terms) are part of their engineering curriculum. 77% of the institutes surveyed said they work with the industry to come up with capstone project ideas, however, overall only 28% of the projects came directly from industry.

Systems Engineering: In a question related to the inclusion of systems engineering in their engineering curriculum a mixed reaction was received. 45% of the surveyed institutions incorporate systems engineering in their engineering curriculum whereas 50% don't and 5% reported that they have been directed to cover some aspects of systems engineering. Systems engineering is defined by[20] as "an interdisciplinary collaborative approach to derive, evolve, and verify a life cycle balanced system solution which satisfies customer expectations and meets public acceptability." Systems engineers evaluate designs using a broader array of measures of effectiveness than simple cost effectiveness.

Though a small sample of the engineering institutions and disciplines were surveyed and even a smaller number responded to the survey they provide useful information in the areas surveyed.

- Engineering design is an important part of the curriculum in all regular engineering programs
- Internship is highly encouraged but not specifically supported
- 90% of the institutions surveyed said they encouraged students to appear for the FE examinations
- Academic and Industry work together to identify Capstone Projects
- Not all institutions have Systems Engineering in their engineering curricula

5. Conclusions

Our conclusions from our review of the literature and our two surveys are:

1. It is extremely difficult to get much detailed information on just what knowledge students in bachelors of software engineering programs are currently being asked to learn, and what basic skills they are asked to be competent in. Although we got a reasonably good response rate to our two surveys (30%), in the case of our main survey only half of the respondents completed the survey, and of those that did, less than half of these took the time to provide detailed information. We conclude that if the consensus of our profession is that we need such information its acquisition must be more formally sponsored.
2. Software engineering's major accomplishment in defining the knowledge and skills that every person who receives a bachelors degree in software engineering should posses, the *Software Engineering 2004: Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering* [10] appears not to be widely known or actively used. Since there is now an effort underway to update these guidelines [21] now is a good time to address this issue. Any of our readers that are interested in this effort should sign up at the referenced site. In addition we hope that this effort hosts forums at several of the significant software engineering conferences, this conference being one of them. We would also suggest that there be some formal coordination/recognition between a Software Engineering 201x Curriculum Guidelines for Undergraduate Degree Programs and the accreditation criteria used by ABET and the professional accreditation organizations in Canada, Australia, and elsewhere.

   On the other hand, about 20% of our respondents did relate their course offerings to the ten SEEK areas so we can conclude that SE2004 is getting some attention. We hope that when the revision is published in a few years that this percentage will more than double.

3. Our main survey did not give us much hard information on how bachelors level software engineering programs compare with the traditional engineering programs. However, with regard to the conclusions stated in the previous section:
   - The SEEK design area received more attention than any of the other areas. Furthermore, since requirements do not typically receive the same amount of attention in traditional engineering as they do in software engineering, if we combine the Software Modeling & Analysis area with the Software Design area the emphasis on design in software engineering compares favorably with that reported for traditional engineering.
   - We did not ask about internships explicitly but from the detailed information we could review from URLs that gave us access to specific programs we feel safe in

asserting that internships are a recognized and tracked component of most software engineering programs.

- In the U.S. at least, software engineering students do not typically take the FE exam as there is only one state that licenses software engineers. We expect this situation to change dramatically in the next decade. See section 3.11 above and conclusion 6 below.
- Again, in our main survey we did not ask about Capstone Projects explicitly but they were cited in open responses and in programs we were given detail information about. Several of the ABET Software Engineering accreditation criteria recommend multidisciplinary projects. We expect that as more software engineering programs come under the purview of engineering departments the percentage of joint academia and industry Capstone Projects will increase.

4. In the light of our remarks above on the significant difference between computer science and software engineering, we were happy to find that nearly 40% of the B.S. in Software Engineering programs are housed in engineering departments. If we assume that all the B.S. in Engineering with a concentration in Software Engineering programs are housed in engineering departments we can estimate that more than half of the bachelors level software engineering programs are housed in engineering departments. In the light of our last conclusion we expect this percentage to increase in the years ahead.

5. We were disappointed in the responses reported above under Industrial Strength Software Development Environments. We probably should have had two questions, one on industrial strength tools and the other on educational applications and materials. We view the use of tools and educational materials as two entirely separate issues, although some times the use of some materials requires the use of specific tools. The home for the effort to revise *Software Engineering 2004* is Ensemble, a new NSF National Science Digital Library [22]. Among other things, Ensemble is attempting to be a gateway to the use, reuse, review, and evolution of educational materials at multiple levels of granularity that supports the entire range of computing educational communities. Not currently linked to Ensemble, but probably should be, is the SEI's Software Engineering Information Repository [23]. We would suggest that the next ASEE conference have a session or workshop on software engineering tools and educational materials repositories and how these are related to the SEEK areas.

6. We were delighted to find that licensure programs for software engineers in an additional nine U.S. states are well underway, and that such programs already exist in Canada and Australia. Although some may find that these early programs are flawed, and that we still have work to do in developing a consensus around what constitutes the core knowledge base and skills in which every software engineer should be competent, we believe that the licensure movement will act a positive forcing function for creating bachelor level software engineering degree programs that accept and address the awesome social responsibilities of software engineers.

Notes

[note 1] Electronic products have similar characteristics, but here the individual physical components and their unique functionality can be isolated.

[note 2] The state of Texas in the United States is the only US state that presently requires licensed professionals to be responsible for software products, and that in only limited venues. In Canada the Canadian Engineering Accreditation Board (CEAB) has accredited three Canadian universities. The graduates of these programs will be eligible for licensing as professional engineers after they have gained supervised experience and pass the usual examinations on law and ethics. The three programs differ greatly. Two were developed with the help of computer scientists; one is very close to a computer engineering program [5].

[note 3] We realize that many academics do not consider Wikipedia references acceptable. For the most part our Wikipedia references are simply to cite definitions. Wikipedia's open editing policy and change logging appears to us to give credence to using Wikipedia definitions as representing community consensus.

[note 4] During the distribution of our survey we found that work on revising *Software Engineering 2004* is already underway[22]. The need for updating *Software Engineering 2004* will probably become more acute as current work on creating a software engineering Professional Engineer examination proceeds[16].

[note 5]. Unless we could not identify a recipient, we sent a message to each of the institutions currently on the ABET web site as having accredited software engineering programs. Had we been able to locate similar accreditation information for Canadian and Australian institutions we would have included them. As it was we probably found many of them through other sources.

[note 6] An angry email complaining about the amount of effort it would take to relate his institution's courses to SEEK was forwarded to one of the authors. Without offering any specifics this recipient asserted that the software engineering courses at his institution did in fact cover all of the SEEK areas. One of the authors also found that some effort was involved in relating his institution's courses to SEEK. The outcomes of all of the software engineering courses in his ABET accredited program are mapped to the ABET criteria, but it is not clear how these criteria related to the SEEK areas. The current effort to revise SEEK [21] might consider coordinating with ABET.

[note 7] From the public record the authors have not been able to tell with certainty what the current official position of the ACM is on the professional licensure of software engineers. This issue is not currently listed in the ACM's public policy page [25] or its U.S. Public Policy Council's (USACM) web page [26]. A search there for "software engineering licensure" did not return anything. The most recent (2002) published statement the authors could find describes the reasons that the ACM was not at that time in favor of licensing software engineers in general [24, 25].

[note 8] The branching logic of the survey only offered this question to those who responded that they had engineering degrees with a concentration in software engineering.

[note 9] Of course formalizing a discipline does not guarantee that such man-made disasters do not occur but formalization does (1) substantially reduce their probability of occurance, and (2) it provides legal ground for the victims of professional error or malfeasance.

[note 10] *Software Engineering 2004, Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering*[10] (SE2004):

> A major challenge in providing curriculum guidance for new and emerging, or dynamic disciplines is the identification and specification of the underlying content of the discipline. Since the computing disciplines are both relatively new and dynamic, the specification of a "body of knowledge" is crucial.

The body of knowledge that is essential for every software engineer to know as "SEEK" – Software Engineering Education Knowledge.

[note 11] We don't mean to slight the existing IEEE-CS certification programs that are based on SWEBOK: the Certified Software Development Associate (CSDA), and the Certified Software Development Professional (CSDP). A great deal of effort has been put into creating, administering, and taking these examinations. This effort demonstrates that at least to some extent software engineering professional certification is valuable. But the CSDA/CSDP exams do not carry much weight in the engineering community at large. Would converting these examinations to software engineering version of the Fundamentals of Engineering (FE) and Principles and Practices in Engineering (PE) examinations and associated licensure be more beneficial to both our profession in general and its individual practitioners?

[note 12] One of our reviewers brought the 1996 SEI report "A Mature Profession of Software Engineering" by G. Ford and N. E. Gibbs [28] to our attention. This comprehensive report identifies eight infrastructure elements of a mature profession; evaluates software engineering circa 1996 against each of these elements; and offers suggestions on what software engineering needs to do to become a mature profession. This paper addresses the circa 2011 state of Ford and Gibbs' first infrastructure element: Initial Professional Education.

Bibliography

[1] Naur, P., and Randell, B., (1968), *Software Engineering: Report of a conference sponsored by the NATO Science Committee*, Garmish, Germany: Scientific Affairs Division, NATO.

[2] Bagert, D. J., (1999), *Taking the Lead in Licensing Software Engineers*, Communications of the ACM, April, 1999, vol. 42, no. 4. 27-29.

[3] Wikipedia, (2011), *Software (disambiguation)*, http://en.wikipedia.org/wiki/Software_(disambiguation), Retrieved 2011-12-23. [note 3]

[4] Wikipedia, (2011), *Engineering*, http://en.wikipedia.org/wiki/Engineering, Retrieved 2011-12-23.

[5] Parnas, D., (2002), *Licensing Software Engineers in Canada*, Communications of the ACM, November, 2002, vol. 45, no. 11. 96-98.

[6] Wikipedia, (2011), *List of Computer Bugs*, http://en.wikipedia.org/wiki/List_of_software_bugs, Retrieved 2011-12-24.

[7] Buckley, F., (1993), *Defining software engineering as a profession*, Computer, August, 1993, 76-78.

[8] Thornton, M., (2009), *Software Engineering PE Examination Development Approved*, IEEE-USA Today's Engineer http://www.todaysengineer.org/2009/Sep/Software-PE.asp, Retrieved 2011-12-24

[9] Thornton, M. & Laplante, P., (2011), *IEEE-USA and IEEE Computer Society Cooperate in New Professional Software Engineering Licensure Initiative*, http://www.todaysengineer.org/2011/Jan/licensure.asp, Retrieved 2011-11-23

[10] The Joint Task Force on Computing Curricula: IEEE-CS and ACM, (2004), *Software Engineering 2004: Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering*, http://sites.computer.org/ccse/SE2004Volume.pdf, Retrieved 2011-12-25

[11] *SurveyMonkey*, (2010), http://www.surveymonkey.com/, Retrieved 2011-12-24.

[12] Wikipedia, (2011), *Computer Science*, http://en.wikipedia.org/wiki/Computer_science, Retrieved 2011-12-25

[13] Wikipedia, (2011), *Software Engineering*, http://en.wikipedia.org/wiki/Software_engineering, Retrieved 2011-12-25

[14] Parnas, D., (1998), "*Software engineering programmes are not computer science programmes*, Annuals of Software Engineering 6, 19-37

[15] McConnell, S., (1999), *Software Engineering is Not Computer Science*, in After the Gold Rush: Creating a True Profession of Software Engineering, (pp 37-43), Redmond, Washington: Microsoft Press.

[16] US NEWS, (2011), *US News: Civil Engineering Rankings*, http://colleges.usnews.rankingsandreviews.com/best-colleges/rankings/engineering-doctorate-civil, Retrieved 2011-11-22

[17] US NEWS, (2011), *US News: Electrical/Electronics/Communications Engineering Rankings*, http://colleges.usnews.rankingsandreviews.com/best-colleges/rankings/engineering-doctorate-electrical-electronic-communications, Retrieved 2011-11-22

[18] US NEWS, (2011), *US News: Mechanical Engineering Rankings*, http://colleges.usnews.rankingsandreviews.com/best-colleges/rankings/engineering-doctorate-mechanical, Retrieved 2011-11-22
[19] Sheehy, K., (2011), *10 Universities With the Most Undergraduate Students***,** http://www.usnews.com/education/best-colleges/the-short-list-college/articles/2011/11/29/10-universities-with-the-most-undergraduate-students, Retrieved 2011-12-07
[20] Blanchard, B. & Fabrycky, W., (2010), *Systems Engineering and Analysis* , Systems Engineering and Analysis, Fifth Edition, Prentice Hall (2010) ISBN-13: 9780132217354
[21] Ardis, M., (2011), *SE 2004 Review Task Force,* http://www.computingportal.org/se2004rtf , Retrieved 2011-12-31
[22] *Ensemble*, (2012), Computing Portal Connecting Computing Educators, http://www.computingportal.org/, Retrieved 2012 -01-04
[23] *Software Engineering Information Repository*, (2012), Software Engineering Institute https://seir.sei.cmu.edu/seir/, Retrieved 2012-01-04
[24] ACM, (2012), *Public Policy*, http://www.acm.org/public-policy, Retrieved 2012-01-02
[25] USACM, (2012), *ACM US Public Policy Council*, http://usacm.acm.org/, Retrieved 2012- 01-01
[26] White, J. & Simons, B., (2002), *ACM's Position on the Licensing of Software Engineers*, Communications of the ACM, November, 2002, vol. 45, no. 11. 91
[27] IEEE Computer Society Professional Practices Committee, *SWEBOK: Guide to the Software Engineering Body of Knowledge, 2004 Version*, http://www.nt.fh-koeln.de/fachgebiete/inf/nissen/softeng/swebok.pdf, Retrieved 2012-3-12
[28] Ford, G. & Gibbs, N. E., *A Mature Profession of Software Engineering*, CMU/SEI-96-TR-004, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, 1996.