Reviews of Curriculum Guides for Professional Software Engineers

Barbara Bernal Thomas
Software Engineering Department
Southern Polytechnic State University

Abstract

Software Engineering (SE) is currently recognized as a stand-alone field of study within the computing disciplines enabling academia's emergence of Bachelor degrees in Software Engineering. This paper examines the recent successful accreditation in the United States of BS in Software Engineering programs by ABET to answer the question of what to teach future professional software engineers. The six accredited program's curriculum is reviewed for similarities and differences. The different licensing views for Software Engineers is presented for insight to what traditional engineering fundamentals should be part of the SE curriculum. The paper begins with the historical evolution of software engineering over the decades. The cornerstones that created the foundations of what we as educators viewed as relevant and current software engineering over the years are explored as a continuum of curriculum progress extending over three decades. The impact and involvement of the SWECC and the SWEBOK project on what we teach in software engineering curriculum are also discussed.

1 Introduction

Since the birth of software engineering at the 1968 NATO conference[19], leading universities have included software engineering within the computing curriculum as topics within courses, sole courses, and recently, as sole degree programs. The demand for skilled software engineering practitioners has had unprecedented growth in industry and academia has had a difficult time keeping up. The first framework for a sole software engineering education was proposed a decade after the 1968 NATO conference. Another decade elapsed before the first model curriculum was designed and the software engineering degree programs began.

Marked by continual change, this last decade has seen steady progress in software engineering education (SEE). In a discipline that is this new, the question of what to teach is particularly difficult to answer; in an innovative field that is drastically changing as quickly as software engineering is, the question of curriculum takes on entirely new dimensions.

In an effort to stay current, the School of Computing and Software Engineering at Southern Polytechnic State University (SPSU) is conducting a curriculum evaluation study of our BS in Software Engineering degree. During the course of the evaluation, we searched the literature for curriculum guidance while gathering material on other software engineering programs currently offered, especially those few that obtained accreditation. Through studying the software engineering education literature, we found various points in the evolution of SE curriculum that shaped the foundations of what educators viewed as relevant and current at the time. For the purposes of this paper, we are calling these points the cornerstones in the evolution of SE curriculum. These cornerstones form a continuum of progress extending over more than three decades that parallels the development of software engineering itself.[22]

One of the cornerstones that are of particular relevance today is the work done by the Software Engineering Coordinating Committee (SWECC) on the Software Engineering Body of Knowledge (SWEBOK) project. Their efforts are investigated along with the current impact of the SWECC and the SWEBOK project on software engineering curriculum. The different views on licensing professional software engineers and different accreditation criteria will also be examined.

Here we present the initial findings of our study to help other institutions in similar situations by providing a concise, although not totally complete, history of how software engineering curriculum has evolved over the decades and examining where it appears to be heading. Our hope is that this paper may serve both to give an overview of SE curricular issues as well as to jump-start the investigation for other schools considering adding a software engineering degree to their program.

2 Early History of Software Engineering Education

Proposed by Peter Freeman et al.[6], the earliest framework for SEE was identified in 1976 as a set of criteria that any SE curricula must follow. The paper defined the following set of five content areas necessary for any SE degree: computer science, management science, communication, problem solving, and design. Emphasis was also placed on the need to incorporate both management and technical issues in software engineering.

Revisiting SEE a decade later, Freeman[7] reported that few, if any, efforts since his earlier paper had "strategically addressed the question of where SEE is or should be headed." He further noted that in spite of the past ten years of development in software engineering, it is not an established part of the educational scene, nor was he aware of any master's-level degree programs in SE at a major university. He did cite the workshops supported by the Software Engineering Institute (SEI) as a beginning for change. While still affirming his initial foundation for SEE, he added that design must have an increased emphasis in SEE.

3 A Specification for Software Engineering

In February 1986, the participants of SEI's Software Engineering Education Workshop revised an initial version of relevant subject areas for a graduate SE degree. The resulting report published in May 1987 was titled Software Engineering Education; An Interim Report from the Software Engineering Institute.[5] The curriculum specification identified twenty content units. For each unit, topics were identified, aspects of those topics were listed, and educational objectives were given. The interim report identified curriculum content without focusing on organizing those topics into courses. The following section describes the efforts in curriculum design that resulted in the first model SE curriculum based on the interim report.

In February 1988 the SEI held a Curriculum Design Workshop[2, 3] to design a curriculum for an MSE degree based on the specification given in the interim report. The task was to partition the identified topics into courses. The committee studied the twenty content units in the specification and identified five subject areas that naturally divided the topics: Systems Engineering, Software Design and Specification, Implementation, Verification and Validation, and Control and Management. Acknowledging that these five subject areas resembled the phases of the traditional waterfall life-cycle model, the committee concluded that the five areas were "legitimate as well as convenient partitions of the curriculum content".[2]

The committee noted that while the courses parallel the phases of a traditional waterfall life-cycle model: requirements, specification, design, implementation, and testing (plus project management), they did not use that as the basis for their design. Rather, the partitioning emphasized the different skills required of the students. They reinforced the idea that while schools would probably order the courses following the life-cycle chronology, there are no prerequisite relationships among the required courses.[22]

4 Computing Curriculum 1991

Computing education as a discipline has grown to encompass various sub disciplines related to computing. In an effort to provide guidance for all computer related fields, the Association of Computing Machinery (ACM) and the Institute of Electrical and Electronic Engineers (IEEE) jointly introduced the broad term "computing" to encompass all of the related disciplines. They published a broad set of curriculum guidelines, Computing Curriculum 1991 [24, 25], that could be applied to any "computing" program.

Computing Curriculum 1991 identifies nine subject areas and three processes in computing. The areas are: algorithms and data structures, architecture methods, artificial intelligence and robotics, database and information retrieval, human-computer communication, numerical and symbolic computation, operating systems, programming languages, and software methodology and engineering. The three processes are theory, abstraction, and design. The software methodology and engineering area has the following five sub-areas: fundamental problem-solving concepts, the software development process, software requirements and specifications, software design and implementation, and verification and validation. Computing Curriculum 1991 does not present a detailed curriculum design, which leaves issues of depth and breadth of coverage of the sub-areas still open. However, it did reinforce the prevalent SE life-cycle

curriculum model since the areas were similar to the five subject areas in the first MSE model curriculum.

5 Software Engineering Body of Knowledge

In 1993, the Joint IEEE Computer Society and ACM Steering Committee for the Establishment of Software Engineering as a Profession were created. In 1998, this committee was superseded by the Software Engineering Coordinating Committee (SWECC) and was established to act as a "permanent entity to foster the evolution of software engineering as a professional computing discipline"[10]. It was determined that achieving consensus by the profession on a core body of knowledge was crucial for the evolution of software engineering to a profession. With the approval of both IEEE-CS and ACM, SWECC set up the Guide to the Software Engineering Body of Knowledge (SWEBOK) project[10] to identify that subset of the body of knowledge that is "generally accepted". The 2004 Guide to the Software Engineering Body of Knowledge is the current edition adopted for use in educational community.

SWEBOK is a project with three scheduled reports: Straw Man, Stone Man and Iron Man. Each report builds upon the previous one. Achieving consensus by the profession has been facilitated by the open solicitation of reviews of the working reports.[24]

The initial Straw Man report selected the Knowledge Areas (KAs) based on "recognized, public and verifiable sources of information" including "tables of contents of general software engineering textbooks, the curricula of undergraduate".[10] The Straw Man KAs consisted of: Development Process, Maintenance Process, Configuration Management, Quality Assurance, Verification and Validation, Improvement Process, Software Development Methods, Software Development Environments, Software Engineering Overview and Definition, Measurement/ Metrics, and Software Reliability.[10] Potential knowledge areas were refined based on the ISO/IEC 12207 standard on Software Life-Cycle Processes. Thus, as in the earlier cornerstones we have presented, the Straw Man knowledge areas still represent a life-cycle model.

The Stone Man report is divided in KA sections that include the knowledge area identification and specification including breakdowns, rationale, standards, and relevant ties to the seven related disciplines above. One major shift in direction that can be seen by examining the progression of the Stone Man documents is a gradual departure from the life-cycle model. There is a continuum of refinement of the knowledge areas away from partitioning based on phases of the life-cycle, to a new, continually evolving software engineering body of knowledge. It is evident that the Stone Man report extends the SE model beyond the influence of the ISO/IEC 12207 standard.[11]

The Iron Man report was conducted in two major sub-phases. Sub-phase 1 during the 2000 to 2002 had goals for experimentation, trial usage, promotion, and development of "performance norms" of this guide. Sub-phase 2 goals were development of the Iron Man version of the Guide

based on the feedback gathered in sub-phase 1 and an extensive review process similar to the review process for the Stone Man phase.[12]

6 Accreditation Initiatives

In order to understand the accreditation initiatives for software engineering, it is worthwhile to examine the accrediting agencies here in the United States. The Accreditation Board for Engineering and Technology (ABET) is a federation of 28 professional engineering and technical societies. Representatives from these societies, who are practicing professionals from industry and academia, form the body of ABET through its Board of Directors and four working Commissions: Engineering Accreditation Commission (EAC), Technology Accreditation Commission (TAC), Computing Accreditation Commission (CAC), and Applied Science Accreditation Commission (ASAC).[22] Software Engineering programs are within The Engineering Accreditation Commission (EAC)

The Computing Sciences Accreditation Board (CSAB) was established to provide for accreditation of post-secondary baccalaureate programs that prepare students for entry into the computing sciences professions. The two member societies of CSAB are the Association for Computing Machinery, Inc. (ACM) and the Institute of Electrical and Electronics Engineers, Inc. -- Computer Society (IEEE-CS). CSAB is the lead society for EAC in ABET, EAC currently accredits program in Software Engineering. The program criteria for the curriculum is:

"The curriculum must provide both breadth and depth across the range of engineering and computer science topics implied by the title and objectives of the program.

The program must demonstrate that graduates have: the ability to analyze, design, verify, validate, implement, apply, and maintain software systems; the ability to appropriately apply discrete mathematics, probability and statistics, and relevant topics in computer science and supporting disciplines to complex software systems; and the ability to work in one or more significant application domains."[4]

Four programs were granted accreditation in 2003 and two more programs were granted in 2004. They are:

1) Clarkson University in Potsdam, NY
2) Milwaukee School of Engineering (MSOE) in Milwaukee, WI
3) Mississippi State University (MSSTATE) in Mississippi State, MS
4) Rochester Institute of Technology (RIT) in Rochester, NY
5) Florida Institute of Technology (FIT) in Melbourne, FL
6) University of Texas (UTA) in Arlington, TX

A study of all six prescribed curriculums is included in the following table organized by common similar courses.[13,14,15,16,17,18]

	Clarkson	MSOE *	MSSTATE	RIT*	FIT	UTA
The following are included in all 6 programs.						
Programming OR Computer Science	1	1	2	3	3	1 1(OO)
Software Engineering	1	1	1	1	1	1
Algorithm & Data Structure	1	1	1	1	1	1
Senior Design OR Project Software Development Lab	1	2 3	2	2	 2	 2
The following are included in 5 out of the 6 programs:						
Discrete Math		1	1	1	1	1
Operating System	1	1	1		1	1
The following are included in 4 out of the 6 programs:						
Computer Arch / Organization	1	1		1		2
Database	1	1	1			1
The following are included in 3 out of the 6 programs:						
App. Domain / Upper Elective		3	3	3		
Ethics /Legal /Social		2	1		1	
Statistic / Probability	1	1		1		
Requirements & Specification		1		1	1	
Human Factors			1	1	1	
Software Verification	1	1			1	
Software System Architecture	1		1	1		
Formal Methods		1		1	1	
Digital Design	1		1(devices)			1(logic)
Programming Lang Concepts	1			1		1
The following are included in 2 out of the 6 programs:						
Software Design	1	2				
Software Components	1	1				
Intro Engineering Concepts		1				1
Algorithms		1	1			
Computer Networks	1	1				
Assembly Language				1	1	

* QUARTERS 1, 2, 3 = # of required courses in the program.

The six programs had single instances of curriculum listed below:

1) Clarkson: Fundamental of SWE - Microprocessors - Linear Circuits[13]
2) MSOE: Management in the Era of Rapid Technology Change - Embedded Systems Software - Embedded Computer System Design - Combinational and Sequential Logic[15]
3) MSSTATE: Project Management - Computer Security - Data Communications - Microprocessor - Distributed Client Server[16]

4) RIT: Students choose 5 out of the following 6: Concurrent Software System – Distributed Software System – Information System Design – Software Process & Product Metrics – Software Verification & Validation – Software Engineering Process[17]
5) FIT: Metrics - Computing & Careers[14]
6) UTA: Compilers - Theoretical Computing[18]

7 Canada's Accreditation Initiatives

Canada began to grant accreditation to software engineering programs in June 2001 by the Canadian Council of Professional Engineers (CCPE). That important milestone was thought to be the road for incorporating new disciplines of engineering into their regulatory system. CCPE's Canadian Engineering Accreditation Board (CEAB) using its existing Accreditation Criteria and Procedures evaluated the software engineering programs for accreditation. Incorporating software engineering into the Canadian accreditation structure has proven to be a learning experience for the profession. The major point of contention between the profession and the Association of Universities and Colleges of Canada (AUCC) was on the appropriate use of the term software engineering in the undergraduate university community. Just over a year before, the Panel on Software Engineering - formed in September 1999 as part of the settlement that saw CCPE end its legal action against Memorial University of Newfoundland over the school's use of the term software engineering in the name of a computer science program - tabled its final report. It recommended the establishment of a new Software Engineering Accreditation Board (SEAB) to accredit all undergraduate software engineering programs offered in Canada; that universities should only use the name software engineering for programs for which SEAB accreditation would be sought; and that names such as software design or software science should be used to describe programs for which SEAB accreditation would not be sought.[22]

The criterion used by the Council was thus completed by the descriptions of various software engineering bachelor degree programs or program options from universities across Canada, be they in Science Faculties or in Engineering Faculties. The Council worked with these documents and tried to find a common denominator. What was reached was a consensus of the council members in the form of a list of ten software engineering components as follows:

1) Requirements and specifications
2) Principles of software analysis, design and architecture
3) Design for embedded, real-time or distributed systems
4) Design and evaluation of user interfaces
5) Software construction
6) Documentation, tools, components, etc.
7) Management of the software process, including metrics and maintenance
8) System performance

9) Quality assurance, testing

10) Software safety

Registration as a professional has choices in Canada with two categories of software professionals: Professional Engineer (P.Eng.) and Information Systems Professional (I.S.P.). Professional Engineering is a self-regulated licensed profession that is governed by provincial law through the Professional Engineers Act. The Information Systems Profession is a self-regulating profession administered by provincial professional societies. Satisfactions of certain academic and work-experience requirements are required for both categories.[22]

8 Licensing of Professional Software Engineers

The curriculum studies presented thus far have illustrated the evolution of software engineering curriculum over the last three decades. The distinction between software engineering and professional software engineering is still fuzzy, and the role that licensing will play is still under debate. Currently the Texas Board of Professional Engineers is licensing professional software engineers. The Association of Professional Engineers and Geoscientists of British Columbia are registering software professional engineers, and the Professional Engineers of Ontario has announced requirements for licensing.[12]

As specified initially, the SWEBOK project had five objectives. However, many believe the fifth objective: to "provide a foundation for curriculum and individual certification and licensing material" was given much more emphasis than originally intended. Concerned about the direction SWECC was moving, ACM established task forces to investigate the issue of licensing software engineers.[1] The study found an "explicit and intimate link" between the SWEBOK project and "the intent and expectation for software engineering licensing".[1] Based on the study, the ACM Council decided in May 1999 that it could not support licensing of software engineers. ACM's rationale was that current software engineering practice is too immature to warrant licensing, and that licensing would not provide assurances about software quality and reliability.

In May 2000, the ACM Council decided that the framework of a licensed professional engineer, which was originally developed for civil engineers, "does not match the professional industrial practice of software engineering. Such licensing practices would give false assurances of competence even if the body of knowledge were mature; and would preclude many of the most qualified software engineers from becoming licensed".[20] Because SWECC became so closely linked with licensing of software engineers, the ACM Council decided to withdraw from SWECC.

The impact that this decision will have on software engineering curriculum is yet to be seen. The distinction between the software engineering body of knowledge and curriculum was discussed in the Straw Man report. It made a clear distinction between the two, stating there were clearly things software engineers must know outside of software engineering. However, the body of knowledge identified in the project should form the core of software engineering

curriculum. Thus it follows that if there is a move towards licensing software engineers under the rubric of the Professional Engineers Licensing structure and requirements, there would be a definite impact on the curriculum of software engineering programs. The examination for licensing professional engineers "would require examinations over subjects most software engineers neither study in their formal education nor need in order to practice software engineering"[1]. Specifically, in addition to computers and math, the fundamentals of engineering examination covers chemistry, ethics, statics, dynamics, electric circuits, and thermodynamics. [23]

The SWECC represented a joint effort involving both IEEE-CS and ACM working together to develop a core body of software engineering knowledge and to foster the evolution of software engineering to a profession. The recent withdrawal of ACM from the SWECC due to differences in opinion on the emphasis being placed on licensing software engineers has abruptly changed the perceived value of the contributions of SWEBOK. As the assessment report states, "Overall, it is clear that the SWEBOK effort is structurally unable to satisfy any substantial set of the requirements we identified for bodies of knowledge in software engineering, independent of its specific content".[20] Due to its significance to SEE, we view this latest development as another cornerstone in the evolution of software engineering curriculum.

10 SPSU Software Engineering 2004 (SE2004)

Section 5 of this paper related efforts made by ACM and IEEE in 1993 for Software Engineering as a Profession. In 1998, these groups created the Software Engineering Education Project (SWEEP) for software engineering curriculum issues at the undergraduate level. The members of SWEEP developed a draft set of accreditation guidelines for software engineering published in the IEEE-CS Computer April 1999 issue. A volume of the computing curricula series dedicated to Software Engineering was the goal. A new SE2004 Steering Committee was formulated in 2001 with additional international members from the Australian Computer Society, the British Computer Society, and the Information Processing Society of Japan. Committee members came from a number of different countries including Australia, Canada, Israel, Japan, the United Kingdom, and the United States. Their mandate was the SE2004 Volume (previously referred to as the Computing Curriculum Software Engineering Volume). Most importantly, the SE2004 project depended on the work of hundreds of international volunteers with interest in improving the quality of undergraduate software engineering education.[21]

The construction of the SE2004 Volume centered around three major efforts:

- Development of a set of desired curriculum outcomes and a statement of what every SE graduate should know.
- Determination and specification of the knowledge to be included in an undergraduate software engineering program (SEEK).
- Construction of a set of curriculum recommendations, describing how a software engineering curriculum, incorporating the SEEK, could be structured in various contexts.[21]

In August 2004 the Software Engineering 2004 (SE2004) was published and made available for public use. It serves as curriculum guidelines for undergraduate for degree programs in Software Engineering. The 135-page volume starts with historical software engineering material parallel to this paper's beginning section and contains in depth specific content for SE courses.[21]

11 SPSU Software Engineering Evolution

The School of Computing and Software Engineering at Southern Polytechnic State University received approval for a Bachelor's of Software Engineering (BSSwE) degree. The proposal sent to the Board of Regents included a preliminary curriculum for the new program, and we completed a curriculum evaluation study to refine the new degree before it was initially offered. The past Fall 2003 marked the first BSSwE graduates.

A large part of our evaluation and related discussions was significantly linked to the current recipients of accreditation and the status of the licensing issue. Since few schools currently offer undergraduate software engineering programs, we want our program to be viewed as a model curriculum that is based on the latest developments in software engineering education. Our concern is that if the licensing trend continues, then we must produce students who will have the requisite background to pass the licensing exam. But on a pragmatic note, we already had to make difficult choices to keep the number of hours required for the BSSwE down to fit the maximum allowed. There simply is no room in our curriculum to include the additional courses in traditional engineering (e.g., statics, dynamics, electric circuits, thermodynamics, etc.) without removing a number of core software engineering courses that cover the common body of knowledge necessary for software engineering. The problem we at Southern Polytechnic State University are facing is one that software engineering educators must face together. Our BSSwE does not include the traditional engineering fundamentals.

This paper presented an on going study, which seeks to answer what to teach future professional software engineers. The software engineering evolution was offered with the major international initiatives towards defining the curriculum for BS in Software Engineering. It examined the cornerstones that shaped the foundations of contemporary SE curriculum by tracing the progress of the last three decades. The current sources for SE education guidance, the SWEBOK and the SE2004 Volume, were reviewed along with the six newly accredited programs. These serve as sources for determination of specific software engineering education. Issues related to software engineering as a profession and the licensing of software engineers were discussed and their effects on software engineering curriculum were explored. Hopefully many will find this review a jump-start to their investigation in this arena.

References

[1] "A Summary of the ACM Position on Software Engineering as a Licensed Engineering Profession" (July 17, 2000) http://www.acm.org/serving/se_policy/selep_main.html

[2] Ardis, Mark and Ford, Gary (1989) 1989 SEI Report on Graduate Software Engineering Education. Technical Report CMU/SEI-89-TR-21, Software Engineering Institute.

[3] Ardis, Mark and Ford, Gary (1989) "SEI Report on Graduate Software Engineering Education" in Proceedings of the Software Engineering Education Conference, edited by Gibbs, N., July 1989, Springer-Verlag.

[4] Criteria For Accrediting Engineering Programs, Nov. 1, 2003. http://www.abet.org/images/Criteria/E001%2004-05%20EAC%20Criteria%2011-20-03.pdf

[5] Ford, G., Gibbs, N., and Tomayko, J. (1987) Software Engineering Education: An Interim Report from the Software Engineering Institute. Technical Report CMU/SEI-87-TR-8, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA.

[6] Freeman, Peter, Wasserman, A.I., and Fairley, R. E., (1976) "Essential Elements of Software Engineering Education", in Proceedings of the 2nd International Conference on Software Engineering, pp. 116-122.

[7] Freeman, Peter (1987) "Essential Elements of Software Engineering Education Revisited." IEEE Transactions on Software Engineering, SE-13, pp. 1143-1148.

[8] Gibbs, N. E., Ardis, M. A., Habermann, A. N., Tomayko, J. E. The Carnegie Mellon University Master of Software Engineering Degree Program, SEI Conference 1990.

[9] Garlan, D., Brown, A., Jackson, D., Tomayko, J., and Wing, J. (1995). "The CMU Master of Software Engineering Core Curriculum" in Proceedings of the 8th Software Engineering Education Conference, edited by Ibrahim, Rosalind, April 1995, Springer-Verlag, pp. 65-86.

[10] Guide to the Software Engineering Body of Knowledge - A Straw Man Version - September 1998.

[11] Guide to the Software Engineering Body of Knowledge – A Stone Man Version (Version 0.6) SWEBOK February 2000. http://www.swebok.org/documents/stoneman06/

[12] Guide to the Software Engineering Body of Knowledge - A Iron Man Version – May 2004 Version. http:/www.swebok.org

[13] http://www.clarkson.edu/

[14] http://www.fit.edu/

[15] http://www.msoe.edu/

[16] http://www.msstate.edu/

[17] http://www.rit.edu/

[18] http://www.uta.edu/

[19] Naur, P., and Randell, B., (1969). Software Engineering: A Report on a Conference Sponsored by the NATO Science Committee, NATO.

[20] Notkin, D., Gorlick, M., & Shaw, M. (May 2000) An Assessment of Software Engineering Body of Knowledge Efforts.

[21] The Joint Task Force on Computing Curricula - IEEE Computer Society and Association for Computing Machinery. (August 23, 2004). Software Engineering 2004. http://sites.computer.org/ccse/SE2004Volume.pdf

[22] Thomas, B. B., Duggins, S. L. (July 2002) "The Internationalization of Software Engineering Education" 2002 American Society for Engineering Education (ASEE) Montreal, Canada.

[23] Tripp, L.L. (March 22, 1999) Professionalization of Software Engineering: Next Steps

[24] Tripp, L. & Frailey, D. J. (Feb. 2, 1999) IEEE Computer Society and ACM Software Engineering Coordinating Committee (SWECC) Overview.

[25] Tucker, Alan B., (Editor) et al., Report of the ACM/IEEE-CS Joint Curriculum Task Force. http://www.acm/education/curr91/homepage.html

BARBARA BERNAL THOMAS

Barbara Bernal Thomas is a full professor in the Software Engineering Department at Southern Polytechnic State University (SPSU) for the last twenty years. The areas of Software Engineering, User-Centered Design and

Computer Graphics & Multimedia are Prof. Thomas' focus endeavors. Barbara is directly involved in the Usability Research Lab at SPSU. She does specialized software development and computer educational support for local businesses in the Atlanta area as a consultant.