AC 2007-1893: IS SOFTWARE ENGINEERING INHERENTLY DIFFERENT THAN OTHER ENGINEERING DISCIPLINES? A CRITICAL ANALYSIS OF ABET SOFTWARE ENGINEERING CURRICULUM GUIDELINES

Sheryl Duggins, Southern Polytechnic State University

Sheryl Duggins is a Professor of Software Engineering at Southern Polytechnic State University in Marietta, Georgia. She received her Ph.D. degree from the University of Florida, and her M.S. and B.A. degrees from the University of Missouri.

Is Software Engineering Inherently Different than Other Engineering Disciplines? A Critical Analysis of ABET's Software Engineering Curriculum Guidelines

Abstract

Since the 1968 NATO Conference which coined the term "software engineering", software practitioners and educators alike have been fighting an uphill battle over the right to be viewed as engineers. The Association of Computing Machinery (ACM) and the Institute of Electrical and Electronic Engineers Computer Society (IEEE-CS) joined forces to try and come to terms with the question of what exactly is software engineering? From the initial work done by the Software Engineering Education Project (SWEEP) that developed draft accreditation criteria for undergraduate degrees in software engineering (SWE) in 1998, to the Software Engineering 2004 (SE2004) report developed by the joint IEEE-CS/ACM task force which presented detailed curriculum guidelines for software engineering undergraduate degree programs, SWE educators have had the luxury of much needed guidance about what our curricula should look like. The icing on the cake took the form of Accreditation Board of Engineering and Technology (ABET) accreditation of software engineering programs by the Engineering Accreditation Commission (EAC); we finally made the cut and were being recognized as real engineers by the engineering accreditation commission, but at what price? SE2004 did an excellent job of elucidating the underpinnings of all engineering disciplines including software engineering. It also identified a number of differences between traditional engineering and software engineering, including the fact that in SWE the foundations are in computer science rather than in the natural sciences, and also that in SWE the focus is on discrete rather than continuous mathematics. SE2004 is an extremely rigorous report, consisting of 129 pages of program specific knowledge developed by our peers for improving the curricula of undergraduate software engineering programs. But as a discipline in engineering, accreditation by ABET is extremely important for program validation. However, according to ABET, the criteria for accrediting software engineering is the same as for all other engineering disciplines, except for two sentences describing program criteria specific to software engineering. If software engineering is so different than all other types of engineering, should ABET guidelines reflect more of these differences? But the real problem is that educators must choose between the advice of software engineers and the ABET guidelines. This author is positing that perhaps we should not have to make that choice.

This paper will examine relevant developments that have shaped our current understanding of what constitutes software engineering; the distinct nature of the Software Engineering Education Knowledge (SEEK); how the SEEK should affect SWE curriculum development; and current ABET curricular guidelines for SWE programs. Finally, the paper will explore the conflicts that arise when trying to design SWE curricula that satisfy both masters: ABET and SE2004.

History of Software Engineering Education

Peter Freeman et. al.[1] proposed the earliest framework for software engineering education (SEE). It was for graduate software engineering, and it identified a set of criteria that any SE curricula

must follow and a set of five content areas necessary for any software engineering (SE) degree. Revisiting SEE a decade later, Freeman[2] 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, SE was 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.

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, resulting in the report: Software Engineering Education, An Interim Report from the Software Engineering Institute published in May 1987[3]. Presenting curriculum recommendations, this report was generally viewed as the first specification for a professional Master of Software Engineering (MSE) degree. The curriculum specification identified twenty content units, and for each unit, topics were identified, aspects of those topics were listed, and educational objectives were given. But the report identified curriculum content without focusing on organizing those topics into courses.

In February 1988 the SEI held a Curriculum Design Workshop[4,5] to design the first model curriculum for an MSE degree based on the specification given in the interim report. The task was to partition the identified topics into courses. Based on the report, the designed curriculum would have 10 to 12 courses, consisting of six or seven core courses, three or four advanced electives, and the remainder would be used for project work. The committee identified five subject areas that naturally divided the topics: Systems Engineering, Software Design and Specification, Implementation, Verification and Validation, and Control and Management.

In an effort to provide guidance for all computer related fields, the ACM and the IEEE jointly published a broad set of curriculum guidelines, Computing Curriculum 1991,[6,7] that could be applied to any computing program. Computing Curriculum 1991 identified nine subject areas and three processes in computing. The area of interest to SEE was software methodology and engineering, which contained 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 did not present detailed curriculum design, but it did reinforce the work done by the SEI workshop as the areas they decided on were similar to the five subject areas in the first MSE model curriculum. It would be a decade later before the first undergraduate software engineering model curricula was developed and disseminated.

SWEBOK

In 1993, the Joint IEEE Computer Society and ACM Steering Committee for the Establishment of Software Engineering as a Profession was created. In 1998, this committee was superseded by the Software Engineering Coordinating Committee (SWECC) which was established to act as a "permanent entity to foster the evolution of software engineering as a professional computing discipline." 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[8] to identify that subset of the body of knowledge that was "generally accepted." The objectives of the SWEBOK project were to:

- Characterize the contents of the Software Engineering Body of Knowledge;
- Provide a topical access to the Software Engineering Body of Knowledge;
- Promote a consistent view of software engineering worldwide;
- Clarify the place of, and set the boundary of, software engineering with respect to other disciplines such as computer science, project management, electrical engineering and mathematics;
- Provide a foundation for curriculum and individual certification and licensing material.

The SWEBOK project had three reports: Straw Man, Stone Man[9] and Iron Man, each of which built upon the previous one. All three have been completed and the Guide to the SWEBOK 2004 is available at http://www.swebok.org.[10] Achieving consensus by the profession has been facilitated by the open solicitation of reviews of the working reports.[11] The three public versions of the report were reviewed by more than 500 reviewers from 42 countries, making it truly an international product. It represents the collective wisdom of software engineers around the globe identifying and specifying precisely what knowledge constitutes the field of software engineering, which is a unique, diverse, and emerging discipline.

One shift in the direction of the SWEBOK project had to do with the role of licensing software engineers [12]. Of the five objectives specified above, many believed 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. The study found an "explicit and intimate link" between the SWEBOK project and "the intent and expectation for software engineering licensing" [13]. Based on the study, the ACM Council decided in May 1999 that it could not support licensing of software engineers and 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." [14]

Because SWECC became so closely linked with licensing of software engineers, the ACM Council decided to withdraw from SWECC. Regardless, the time for instituting licensing software engineers has already begun. Canada views software engineering as a specialty within engineering that has its own specific knowledge. "The most difficult task facing us in the licensure of software engineers is the identification of the special knowledge that should be required of those who practice in this field." [15] Legislation is currently in place in Canada for licensing software engineers, and some licensed software engineers are practicing today. In the US, Texas leads in the licensing of software engineers. Licensure was based on the common standard used for all fields of engineering and the Texas Board of Professional Engineers was charged by the state legislature with implementing and enforcing the licensing of professional engineers (P.E.s). Since the Texas Board interprets the legislative mandate as including software engineers, those who offer services as software engineers are required to obtain the P.E. license in Texas. This raised the question of whether the rules enforced by the Texas Board seemed to

force software engineers into a mold designed for other distinct disciplines so that a P.E. license could be issued, or whether software engineering was added to the P.E. license to allow the Board's governing regulations to match actual practice rather than to force the practice into a particular mold. The Texas licensing committee requested help from ACM and IEEE-CS to jointly define the body of knowledge on which software engineering licensing was based [16].

Curriculum Evolution and Accreditation

Whereas licensing is the main concern of working and aspiring engineers, accreditation is the main concern of engineering educators. In addition to overseeing the SWEBOK project, the SWECC committee was also charged with coordinating the Software Engineering Education Project (SWEEP), which was responsible for the development of the first draft accreditation criteria for undergraduate software engineering programs.[17] The accreditation criteria for Software Engineering developed in 1998-1999 is available at: http://www.acm.org/serving/se/Accred.htm. These initial accreditation criteria focused primarily on the software engineering curriculum, specifying that "the program must include approximately equal segments in software engineering, in computer science and engineering, in appropriate supporting areas, and in advanced materials. This material should cover about three-quarters of the overall academic program, with the remainder to include institutional requirements and electives." Specific curricular guidance containing both required coursework including requirements, software architecture and design, testing and quality assurance, etc., as well as more general guidelines like programs should include work in one or more significant application domain areas. Unlike the ABET curricular guidelines which appeared years later, these initial software engineering accreditation guidelines made it very clear that software engineering is different than traditional types of engineering and they spelled out precisely how that difference should manifest in curricular design.

The SWEBOK project was only intended to describe the knowledge needed to engineer software, and did not include non-software engineering knowledge needed to practice, nor did it include curricular guidance. The SWEEP committee used the SWEBOK knowledge areas to select the Software Engineering Education Knowledge (SEEK) that would serve as the foundation for an undergraduate software engineering curriculum. This curriculum evolved from the Computing Curricula '91[18], to the Computing Curriculum Computer Science (CCCS)[19]. This evolved further when SEEK volunteers worked together with the CCSE Steering Committee to develop the Computing Curriculum Software Engineering 2001(CCSE 2001)[20], which developed into Computing Curriculum Software Engineering 2004(CCSE 2004), which has become known as Software Engineering 2004 (SE2004)[21].

As stated in SE2004: "This document is intended as a resource for institutions that are developing or improving programs in software engineering at the undergraduate level, as well as for accreditation agencies that need sample curricula to help them make decisions about various institutions' programs."[22] Thus SE2004 was developed by leaders and experts in the field of software engineering as a prescriptive specification of the knowledge to be included in an undergraduate software engineering program.

During the same time period, ABET through their EAC, approved accrediting undergraduate software engineering programs and identified a set of non-prescriptive criteria to be utilized.

The first software engineering programs were accredited by ABET in 2003, and as of today, March 2007, 13 undergraduate software engineering programs have been accredited by ABET.[23]

Is Software Engineering Different Than Other Types of Engineering?

The perceptions of the difference between software engineering and traditional types of engineering can be seen internationally. By definition, SWEBOK and the need to clarify what software engineering is, was an international effort. As was mentioned earlier with respect to licensure, licensing software engineers as professional engineers (P.E.s) has certainly brought attention to the differences in the bodies of knowledge that software engineers hold versus other engineers. To be able to license software engineers requires a focus on that specific knowledge that practicing software engineers hold. In Canada, there have been some problems due to these differences both with accreditation of software engineering programs and with the use of the term engineering in software engineering undergraduate programs. The Canadian Council of Professional Engineers (CCPE) sued Memorial University of Newfoundland in federal court for trademark infringement over the use of the term engineering in a degree program with a specialty in "software engineering"[24]. As a result of that lawsuit, an independent panel was established that proposed a new Software Engineering Accreditation Board (SEAB) for accrediting software engineering programs. The accreditation criteria and procedures of the new board were to be developed jointly by the Canadian Engineering Accreditation Board (CEAB) and the Computer Science Accreditation Council. However, after the two accrediting bodies had jointly drafted an accreditation plan for the SEAB, the CEAB "recommended a number of significant changes to the structure and curriculum requirements of [SEAB] without consulting the [CSAC]. The CCPE subsequently passed a motion supporting the [panel] recommendations based on the amendments from its accrediting arm"[25]. The AUCC concluded that it could not endorse the panel's approach due to "the current state of relations between the engineering and computer science communities"[26]. However, the CEAB has subsequently accredited three software engineering programs, all offered through the universities' engineering faculties.

When ABET began accrediting software engineering programs by the EAC, we (software engineers) heaved a giant sigh of relief – we finally made the cut and were being recognized as *real* engineers by the engineering accreditation commission. But with this recognition, as a discipline in engineering with the *possibility* of accreditation by the EAC of ABET, things changed for undergraduate software engineering programs: as an engineering program, it became extremely important to achieve accreditation for program validation. I ask again: but at what price?

SE2004 clearly elucidated the underpinnings of all engineering disciplines including software engineering and identified a number of differences between traditional engineering and software engineering. The differences in both the foundations and the mathematics areas are germane. Specifically, unlike other engineering disciplines with foundations in the natural sciences, the foundations in SE are in computer science. And in SE the focus is on discrete rather than continuous mathematics. "Discrete mathematics is the mathematics underlying all computing, including software engineering. It has the importance to software engineering that calculus has to other branches of engineering. Statistics and empirical methods also are of key importance to software engineering….SEEK does not contain calculus because it is not used by software

engineers except when doing domain-specific work (e.g. for other engineers or for scientists) and hence is not essential for *all* software engineering programs."[27] Regarding the science courses, SEEK does not specify science, as it falls in the area of "non-SEEK topics". But SE2004 states the following regarding science and engineering courses: "These cover material such as physics, chemistry, electrical engineering, etc. Most software engineering programs, especially in North America, will include some such courses, particularly physics…Taking some science and engineering courses will also help students who later on want to develop software in those domains."[28] Referring back to the initial draft accreditation guidelines developed by the SWEEP, it is interesting to note that the mathematics they prescribed was discrete mathematics and probability and statistics and did not include calculus. They also suggested no specific science classes but stated that some of the domain areas may require additional math and science.

SE2004 is an extremely significant report to software engineering educators that was intended to give program specific guidance developed by software engineers for improving the curricula of undergraduate software engineering programs. But as a discipline in engineering, if we seek accreditation by ABET, then we must follow ABET criteria for accrediting engineering programs rather than SE2004. According to ABET, the criteria for accrediting software engineering is the *same* as for all other engineering disciplines, except for two sentences describing program criteria specific to software engineering. Regarding the curriculum for SE programs, ABET states the following: "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."[29] If software engineering is so different than all other types of engineering, should ABET guidelines reflect more of these differences?

The only other curricular advice specifically given by the ABET Criteria is specified in Criterion 4 – Professional Component. It says that the "faculty must ensure that the program curriculum devotes adequate attention and time to each component, consistent with the outcomes and objectives of the program and institution. The professional component must include: (a) one year of a combination of college level mathematics and basic sciences (some with experimental experience) appropriate to the discipline and (b) one and one-half years of engineering topics…"[30] This appears to be clear enough, with the most important part being "appropriate to the discipline" which, since the SEEK specifies what that knowledge is and SE2004 specifies curricula "to help accreditation agencies… make decisions about various institutions' programs", the two agencies should be working together, with ABET deferring to SE2004 for what *specific* subjects are appropriate to the discipline. But that doesn't seem to be happening.

Of the 13 ABET accredited software engineering programs, all require at least two semesters of calculus and two semesters of calculus-based physics (with most requiring additional advanced classes). Certainly with academic freedom, each program decides what specific courses they choose to offer, but the real problem is what if software curriculum designers don't choose to follow this typical engineering sequence of courses? Consider a software engineering program based on SE2004 without the typical engineering advanced math and science courses, for

example, non-calculus based physics. Nowhere in the ABET criteria nor in the SE2004 does it specify that a SE program must have physics, nor does either say any physics courses included in a SE program must be calculus-based, but that bias has become the norm. The problem is that educators must choose between the advice of software engineers (SE2004) and the ABET guidelines, with the caveat that strict adherence to SE2004 may result in not achieving ABET accreditation. This author is positing that perhaps we should not have to make that choice.

Conclusion

This paper examined the relevant developments that have determined our current conception of software engineering. A brief history of software engineering education was given; the SWEBOK project was discussed; and the evolution of the model curriculum for undergraduate software engineering that resulted in SE2004 was presented. The evolution of accreditation criteria was also reviewed and the role that SEEK and SE2004 played in initial accreditation efforts was explored. Finally the question of whether software engineering is inherently different than traditionally types of engineering was investigated, and how current ABET curricular guidelines for SWE programs makes the distinction. Finally, the paper elucidated the conflicts that arise when trying to design SWE curricula that satisfy both masters: ABET and SE2004.

Bibliography

[1] 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.

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

[3] 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.

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

[5] 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.

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

[7] Tucker, A.M., (1991). "Computing Curricula 1991," Communications of the ACM, 34(6), pp. 68-84.

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

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

[10] Abran, Alain & Moore, James W. (Exec. Eds.), Bourque, Pierre, Dupuis, Robert, and Tripp, Leonard L., (Eds.), Guide to the Software Engineering Body of Knowledge: 2004 Version, IEEE, Inc., February 2004. http://www.swebok.org/documents/stoneman06/

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

[12] Duggins, S. (March 2001) "Curriculum Impact of the Maturing Software Engineering Profession" in Proceedings in the 2001 ASEE Southeastern Section Conference.

[13] ACM Council "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

[14] Notkin, D., Gorlick, M., & Shaw, M. (May 2000) An Assessment of Software Engineering Body of Knowledge Efforts. http://www.acm.org/

[15] Modesitt, Kenneth, L. (February 2002) "International Software Engineering University Consortium (ISEUC): A Glimpse into the Future of University and Industry Collaboration" In Proceedings of the Fifteenth Conference on Software Engineering Education & Training, pp. 32-41, IEEE Computer Society, Los Alamitos, California.

[16] ACM Panel on Professional Licensing in Software Engineering "Report to Council" (May 15,1999) http://www.acm.org/serving/se_policy/report.html

[17] Barnes, B., et. al., "Draft Software Engineering Accreditation Criteria", Computer, April 1998.

[18] Tucker, A. B., et. al., Computing Curricula '91, Association for Computing Machinery and the IEEE Computer Society, 1991

[19] ACM/IEEE-Curriculum 2001 Task Force, Computing Curricula 2001, Computer Science, December 2001. http://www.computer.org/education/cc2001/final/index.htm

[20] ACM/IEEE-Computing Curriculum – Software Engineering 2001 Joint Task Force on Computing Curricula, IEEE Computer Society and the Association for Computing Machinery, Computing Curriculum Software Engineering Public Draft 1, July, 2003.

[21] ACM/IEEE-Computing Curriculum – Software Engineering 2004 Joint Task Force on Computing Curricula, IEEE Computer Society and the Association for Computing Machinery, Computing Curriculum Software Engineering 2004, February, 2004. http://sites.computer.org/ccse/

[22] Ibid pg 51

[23] Accreditation Board for Engineering and Technology, Accreditation Policy and Procedure Manual, ABEC Inc., November 2006. http://www.abet.org/

[24] Lemay, Marie (Sept. 2001) "Accreditation of Software Engineering Programs is Good News" http://www.ccpe.ca/e/pub_ceo_01_02.cfm

[25] Mohr, Jonathan, (May 15, 2002) "Conflict in Canada over Software Engineering", Forum for Advancing Software Engineering Education (FASE), Volume 12, Number 05, Issue (148)

[26] Ibid

[27] Computing Curriculum Software Engineering 2004, pg 57

[28] Computing Curriculum Software Engineering 2004, pg 64

[29] Accreditation Board for Engineering and Technology, Criteria for Accrediting Engineering Programs, ABEC Inc., November 2006, p. 18. http://www.abet.org/

[30] Ibid pg 2