An Assessment of Software Engineering Body of Knowledge Efforts

A Report to the ACM Council

David Notkin (chair)
Department of Computer Science & Engineering
University of Washington
Box 352350
Seattle WA 98195-2350 USA
notkin@cs.washington.edu

Michael Gorlick
The Aerospace Corporation
P.O. Box 92957
Mail Station M1-102
Los Angeles, CA 90009 USA
gorlick@aero.org

Mary Shaw
Computer Science Department
Carnegie Mellon University
Schenley Park
Pittsburgh PA 15213-3891 USA
Mary.Shaw@cs.cmu.edu

May 2000

I. Executive Summary

In May 1999 the ACM Council adopted a resolution both reaffirming ACM's commitment to solving the software quality problem and also stating its opposition to licensing software engineers, on the grounds that licensing is premature and would be ineffective in addressing the software quality problem. ACM Council also affirmed its interest in developing a core body of knowledge for software engineering, which is often viewed as an appealing step toward improving software quality. Our committee was formed to study the existing software engineering body of knowledge efforts — including SWEBOK, with which ACM is involved through the joint IEEE CS/ACM Software Engineering Coordinating Committee (SWECC). Our charge was to determine the status, progress, and likely outcome of these efforts. This report documents our findings, our reasoning, and our conclusions.

Our study and analysis has led us to the conclusion that the current software engineering body of knowledge efforts, including SWEBOK, are at best unlikely to achieve a goal of critical importance to ACM: the ability to provide appropriate assurances of software quality for

software systems of public interest. Although the body of knowledge efforts may possibly make progress towards other stated and unstated objectives, we believe that ACM's continued participation in the SWEBOK effort will not further — and indeed, may distract from — efforts to improve software quality, especially for systems of public interest. It would be consistent with our analysis for the ACM Council to choose to withdraw ACM from further involvement with SWEBOK, largely because of the danger of pursuing a path that might well provide false assurances to the public. Furthermore, we are uncertain whether, at present, there exists any process to articulate a core body of knowledge in software engineering that will directly contribute to the solution of the software quality problem. At a minimum, a conceptually clear and generally accepted organizing principle is a necessary pre-condition to the effective articulation of such a software engineering body of knowledge. No compelling organizing principle exists at present, and we see no clear course of action that would be likely to lead to one in the next few years.

II. Charge to Committee from ACM Council

The ACM is resolved to pursing a core body of knowledge for software engineering as part of its commitment to improving the quality of software. We are asking your committee to assess the current efforts in developing a software engineering body of knowledge (including but not restricted to the ongoing SWEBOK effort, under the guidance of SWECC, the joint committee of the ACM and the IEEE Computer Society). Based on a careful study of these plans and any other relevant material, we request that you determine whether you find that any or all of these efforts are on track to providing a high-quality core body of knowledge for software engineering. If you determine that no effort is likely to produce a high-quality body of knowledge, then please additionally provide advice to the ACM about how to pursue such a body of knowledge.

We understand the "body of knowledge for software engineering" to mean a document that identifies the concepts, facts, and skills that are expected to be mastered by practicing software engineers. The distinction between a "body of knowledge" and a "core" body of knowledge is that the core focuses only on the material within software engineering that is expected to be mastered. Material from related disciplines — such as project management — may also be necessary for software engineers, but is omitted from the core.

III. Brief Background

The Joint IEEE Computer Society and ACM Steering Committee for the Establishment of Software Engineering as a Profession was established in 1993.[1] One of the initial recommendations of this committee was to work towards defining a "required body of knowledge and recommended practices." 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." SWECC, with the approval of the two societies, can set up projects in this general area; a key project of SWECC has been the SWEBOK effort.

[1] This material is taken primarily from the brief history found at http://www.computer.org/tab/swecc/.

In March 1999, ACM named an Advisory Panel on Professional Licensing in Software Engineering to make recommendations to ACM Council about the role that it should take with regard to issues related to the licensing of software engineers.[2] In part based on this report, in May 1999, the ACM Council approved a policy opposing the licensing of software engineers at present "because ACM believes that it is premature and would not be effective at addressing the problems of software quality and reliability." At the same time, the Council stated that the "ACM is, however, committed to solving the software quality problem by promoting R&D, by developing a core body of knowledge for software engineering, and by identifying standards of practice." Thus, as part of the ACM's commitment to articulate a core body of knowledge, Council wanted to understand the current state of affairs of such efforts, which led to the formation of this committee.

Our charge focuses on the software engineering body of knowledge efforts. However, it is impossible to separate our assessment from more general issues related to software professionalism and credentialing, precisely because that has been a primary and explicit motivation for the two efforts that we assess. As we discuss below, our focus will be on the effectiveness of the body of knowledge efforts in improving the quality of software products as they relate to the public interest; we believe this is consistent both with our charge and also with the fundamental interests of the ACM.

IV. Existing Body of Knowledge Efforts

We assessed the two major existing efforts, the SWEBOK effort[3] (http://www.swebok.org) and the Australian Computer Society (ACS) Core Body of Knowledge for Information Technology Professionals (http://www.acs.org.au/national/pospaper/bokpt1.htm). Because of the many similarities between the two efforts, our primary focus in this document is on the SWEBOK effort.

As described above, we understand the "core body of knowledge for software engineering" to mean a document that identifies the software engineering concepts, facts, and skills that are expected to be mastered by practicing software engineers.

- SWEBOK states: "The software engineering body of knowledge is an all-inclusive term that describes the sum of knowledge within the profession of software engineering."
- The ACS effort states: "This report identifies the 'Core Body of Knowledge' in Information Technology which *all* [original emphasis] I.T. professionals practising in Information Systems, Computer Science and Computer Systems Engineering should be expected to have."

SWEBOK states explicitly its intent to include only "generally accepted knowledge" in its body of knowledge. They provide two definitions of "generally accepted":

---

[2] Information on the panel, its members, their position papers, and the report is available on the ACM web site at http://www.acm.org/serving/se_policy/.

[3] P. Bourque, R. Dupuis, A. Abran, J. W. Moore, and L. Tripp. The Guide to the Software Engineering Body of Knowledge. *IEEE Software*, vol. 16, pp. 35-44, November/December 1999.

- The first definition is derived from the one used by the Project Management Body of Knowledge (http://www.pmi.org/publictn/pmboktoc.htm): "`Generally accepted' means that the knowledge and practices described are applicable to most projects most of the time, and that there is widespread consensus about their value and usefulness. `Generally accepted' does not mean that the knowledge and practices described are or should be applied uniformly on all projects; the project management team is always responsible for determining what is appropriate for any given project. "
- "The [SWEBOK] Industrial Advisory Board better defines `generally accepted' as knowledge to be included in the study material of a software engineering licensing exam that a graduate would pass after completing four years of work experience."

Each of these efforts has provided explicit reasons for their development including:

- SWEBOK
    - To help software engineering reach the status of a legitimate engineering discipline and a recognized profession.
    - To improve the level of professional practice.
    - To provide a basis for the development and accreditation of university curricula and the licensing and certification of professionals.
- ACS IT Core Body of Knowledge
    - To assist in the processes of assessing applications for membership of the Australian Computer Society.
    - To aid in tertiary course accreditation.
    - To help assess applications for migration to Australia.
    - To provide guidance for course design and implementation.
    - To contribute to pending legislation in professional standards.

The second definition of "generally accepted knowledge" and the statements of purpose provide an explicit and intimate link between these efforts and the intent and expectation for software engineering licensing.

V. Requirements for a Core Body of Knowledge

To assess the body of knowledge efforts, we identified a set of requirements against which they could be assessed. The reasoning behind this set of requirements is important to understand; this reasoning is intricate, so we present it with some care.

We chose to pursue the identification of these requirements by starting from first principles. In particular, we considered the question, "Where do the interests of the ACM lie with respect to these issues?" We identified three specific interests.

- First, ACM has an interest in software as it affects the broad public interest including matters of safety, economic and social consequences, and quality of life: in other words, software as it touches and influences the public at large.

- Second, ACM has an interest in the professional development of its members as the practitioners and standard bearers for a technology that is reshaping society in as fundamental a way as the Industrial Revolution of a century ago.
- Third, ACM has an interest in the development and maturation of the computing disciplines, both theoretical and practical, including software engineering.

Clearly, the controversial issue of professional registration and credentialing falls squarely within the ACM's range of interests. Professional registration is one of many forms of credentialing ranging from public requirements and exams (such as the state bars for lawyers or board certification for medical doctors) to private certification (such as certificates issued by a specific vendor or administered by an industry association).

Each specific form of credential offers particular assurances regarding the breadth of knowledge, depth of practice, and degree of competence of those who receive the credential. Engineering registration, like that for law or medicine, is held to a high standard. Since it is managed in the public interest it implies that its holders will practice at a level consistent with public safety (but does not imply perfection). In contrast, the product-specific certifications managed by vendors imply proficiency in the use of certain products, but nothing more.

A document (or more properly a comprehensive collection of materials) that articulates the concepts, facts, and skills required of a competent software engineer, backed by the requisite reference works would serve all three of the interests of the ACM, especially the professional development of its members. Such a document —a software engineering core body of knowledge —would be challenging to articulate and maintain. Like comparable engineering or medical standards of practice and technology it must distinguish with precision and care those elements of the knowledge base that are routine and expected to achieve good practice from those that are untried, untested, or unrefined, awaiting practical, comprehensive validation.

Furthermore, software development involves many tasks and roles, of which only a subset carries the burden of professional engineering (for example, an analysis by an engineer for the purpose of attesting to software safety versus the development or testing of a single function in a software system). In this light a body of knowledge for software engineering must identify the various roles within a software project and explicitly enumerate the knowledge, skills, and practices required for each. Such a body of knowledge emphasizes the additional knowledge required of a professional engineer beyond the knowledge that all competent software practitioners are expected to have.

The breakneck pace of change in software and related technologies ensures that any body of knowledge, no matter how constituted or structured, will be obsolete in critical portions the day that it is published. Hence, like any well-designed software system, mechanisms must be in place to ensure that the body of knowledge undergoes continual review and revision in light of advances in theory and practice.

What role does a body of knowledge play in assuring that software protects the public interest? Software quality can be established and validated in at least three ways:

- by attending to the product (of which software may be only a part);
- by attending to the organization producing the product; or

- by attending to the individuals building the product.

Standards for products and organizations are the legitimate concern and topic of investigation of a variety of organizations.[4] Product-oriented examples include the Internet Engineering Task Force and the World Wide Web Consortium, which establish standards and practices for numerous Internet protocols and applications. Organization-oriented examples include the Software Engineering Institute and ISO, which provide methods for evaluating the capabilities of organizations developing software products.

A body of knowledge for any role in software development would address the individuals involved in that role of software systems development— the members of the organizations constructing the software products. Thus with respect to software quality the individuals are only a part (albeit an important one) of the story of software development, deployment, and use. In particular, a software engineering body of knowledge would be expected to address the knowledge of individuals involved in the engineering roles for software.

In addition to the complex relation among the product, the producing organization, and those individuals developing the software, the form, requirements, and substance of the assurance vary with the domain and intent of the product.

Software is of increasing public importance, both as an essential element of engineered systems and as the principal embodiment of capabilities whose failure is of nontrivial consequence to the public at large or to large groups of individuals. The public desires and deserves assurances about the quality both of systems with embedded software and also of systems that are principally embodied in software where their failure is of nontrivial public consequence. The exemplars of systems with embedded software in which the public has demonstrated interest include such safety-critical systems such as airplanes and nuclear power plants. Examples of more "pure" software systems in which the public has interest include many systems used by the Internal Revenue Service and the Social Security Administration. For such systems, publicly governed credentials and assurances are appropriate, if achievable.

It is essential to distinguish the public interest from private interests. Software is of increasing private importance; by private we mean that successes or failures of the software systems may be of serious consequence to those involved, but the public interest is not generally at issue. For example, for many large integrated systems (say shipping and receiving or industrial process control) final responsibility for assurance rests with the system engineer; the software component of that system is of private interest. An example of a private failure of software that is visible to the public but is not of public interest might be the inability of an e-commerce web site to allow people to order the items that they want. The company may fail because of this, but the general public does not have any more interest in this situation than it does in any company (software-based of otherwise) that fails because of its inability to meet customer expectations.[5]

---

[4]As an example outside of computing, for pharmaceutical development the FDA has chosen to attend to the product by demanding extensive and costly protocols for testing new drugs rather than relying on recognizing and credentialing the scientists who design those drugs as part of a profession.

[5] The current state of warranties of software products is highly unsatisfying and deserves attention. However, consistent with the reasoning in the body of this report, we see no reason to believe that developing a body of knowledge will improve this situation.

A body of knowledge can in principle serve individual engineers by guiding their professional study or by serving as a basis for credentialing. A variety of credentials could be based on a body of knowledge. Appropriate credentialing (public versus private, generic versus product-specific, general versus domain-specific) depends on the character of the software systems at issue and whether achievable practice is adequate for the implications of the credential.

Thus a body of knowledge alone may be inadequate or ill suited for some forms of credentialing or may require augmentation with domain-specific content. Specifically, a body of knowledge may take on quite different forms in response to the kinds and degrees of the assurances provided by the credential.

To succeed as a basis for credentialing, a body of knowledge must command the respect and trust of the community it serves. The quality of a body of knowledge can be ensured using the same processes that we use for software — by direct evaluation of the product (the body of knowledge itself and the supporting materials), by the credibility of the institutional process that created the product (the selection of contributors, the review cycle, the procedures for recording, tracking, and correcting errors or omissions), and by the credibility of its authors, editors, and other contributors. For example, the body of knowledge represented by the RFCs of the Internet Engineering Task Force, commands substantial respect and trust since the product (Internet standards and practices) are comprehensive and of high quality, the institutional process is open, fair, and well understood, and its authors and editors possess unquestioned expertise. The IETF is also highly responsive and adaptive, effectively managing the exceedingly rapid changes to the Internet changes. The IETF remains relevant precisely because it has adapted so quickly and so well. Overall, the IETF is a standard of excellence to which any body of knowledge might aspire.

The field of software engineering changes rapidly as well — in content, technology, and the assimilation of new techniques. Therefore a body of knowledge must embrace and institutionalize means for evolving in response to changing needs, technology, and knowledge.

From the issues outlined above we can extract requirements for a software engineering core body of knowledge:

- It must reflect actual achievable good practice that ensures quality consistent with the stated interest; it is not that following such practices are guaranteed to produce perfect software systems, but rather that doing so can provide reasonably intuitive expectations of quality.
- It must delineate roles among the participants in a software project.
- It must identify the differential expertise of specialties within software engineering.
- It must command the respect of the community.
- It must embrace change in each and every dimension of its definition; that is, it must be associated with a robust process for ensuring that it is continually updated to account for the rapid change both in knowledge in software engineering and also in the underlying technologies.

These requirements define criteria for assessing body of knowledge efforts.

VI. Assessment of SWEBOK

To what degree does SWEBOK satisfy the requirements laid down in the previous section? Overall, it does not fare well.

- Although SWEBOK states that "generally accepted" knowledge is the chief criterion for inclusion of material, the selection process for including topics did not account for what is achievable good practice. The SWEBOK web site describes how the knowledge areas were chosen for the initial SWEBOK Strawman effort:

  > "In order to propose Knowledge Areas and Related Disciplines for `generally accepted' knowledge and to do so based on recognized, public and verifiable sources of information, it was decided that the tables of contents of general software engineering textbooks, the curricula of undergraduate and graduate programs in software engineering and the admission criteria for graduate programs would constitute the input to our analysis. A total of 24 textbooks and 29 programs were examined. For the purposes of this Straw Man version, a potential knowledge area had to be mentioned in the table of contents of at least one quarter of the textbooks sampled to qualify as a proposed Knowledge Area."

  It is conceivable that this approach links to actual practice, but if so it is the responsibility of SWEBOK to show that these textbooks appropriately reflect actual practice. It is our opinion that there is usually a gap — indeed, often an enormous gap — between actual practice and what appears in textbooks.

- SWEBOK does not distinguish among possible roles within a software development effort. This does not reflect the most common practices in the field, where these distinctions are common (although they vary from company to company and project to project). Different roles require different skills, which rely on different knowledge. Furthermore, not all roles (and indeed not all roles currently called "software engineering") involve responsibilities commonly regarded as engineering responsibilities.

- SWEBOK has not addressed the notion of bodies of knowledge for specialty areas, so it has not considered representations of that knowledge. Indeed, SWEBOK is explicit in focusing on generally accepted knowledge while not considering specialty areas in which there are "practices used only for specific types of software."

- It is not yet clear whether SWEBOK would carry authority, although this is clearly a central goal of the SWEBOK team as reflected in their literature as well as in their drive to include many organizations (including ACM) in the effort. At the same time, our understanding is that the companies represented on the SWEBOK Industrial Advisory Board must make substantial financial contributions to the SWEBOK effort. Although this may of help to ensure that SWEBOK is properly funded, it almost certainly disenfranchises some companies and significantly harms the potential authority that SWEBOK might otherwise hold. The credibility and authority of the individuals who participate in the effort will also have a significant effect on the credibility and authority

of the product.

- A process for updating the SWEBOK results would surely be based on the initial SWEBOK development effort and would be unlikely to succeed.

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.

VII. Observations and Conclusions

The issues surrounding software engineering bodies of knowledge and the entire discussion of "software engineering as a profession" are extremely subtle and complex. Many dedicated and talented people have spent, and will continue to spend, significant effort in trying to address these issues. We believe very strongly that the fundamental goal of people in the community is shared: there is a great and legitimate societal and economic need to improve our ability to effectively engineer software systems.

However, it is clear that there are deep and consequential differences in how we should proceed towards that goal, at least in some key dimensions.

- A common assertion is that SWEBOK and other activities related to software engineering as a profession are crucial as drivers that will lead to essential improvements in our ability to engineering high-quality software. That is, even if we are not yet able to lay down an effective body of knowledge, trying to do so will further our insights and understandings in ways that will bring us to this goal much more quickly.

  This is a serious and legitimate point of view — indeed, it was the primary argument used to encourage ACM's involvement in this process. We agree that there is potential benefit to articulating important knowledge in a manner that can help professional development and thereby the state of practice. Our concern is the clear link between the body of knowledge efforts and the haste to start credentialing based on such bodies of knowledge.

- A related assertion is that involvement SWEBOK and other related activities is necessary as a reaction to the Texas move towards licensing and potential similar moves in other regions. The justification is based on the observation that Texas is going to license software engineers, so it is important to influence the process in as positive a way as possible. It is, of course, accurate that Texas is licensing software engineers; indeed, roughly two dozen have been licensed to date.

  We believe, however, that this view is an inappropriate one since we do not believe that we can yet identify an achievable level of professional practice that will provide appropriate assurances of software quality to the public. Absent an adequate, achievable professional level of practice, it is not sensible to test it. Providing false assurances to the public — even when directed to do so by a governmental agency —is at best

inappropriate.

- The SWEBOK effort uses the notion of "generally accepted knowledge" as a cornerstone, specifically excluding "practices used only for specific types of software."

  We believe very strongly that, for software, this is approach is highly likely to fail and that the opposite approach — primarily focusing on specific domains — is far more likely to succeed. The central reason is that software engineering addresses a much broader scope than traditional engineering disciplines. That is, the differences among software systems are extremely large: consider the widely varying issues that must be addressed in handling even just a few kinds of software systems such as software driving a digital watch, the software driving Internet routers, the software driving a radiation-treatment planning system for oncologists, the software driving the database for online electronic retailers, and the software driving logistics planning for a large military organization. The content required to characterize this diverse universe is better broken down by domain (that is, by approaches shared by "specific types of software"). As a central example, which is also consistent with our focus on the public interest, safety-critical systems might be a promising domain for defining a specialized body of knowledge.

- We are not certain as to what balance of attending to the product, to the organization producing the product, and the individuals building the product will be most effective for improving software quality. We are, however, certain that this balance must be achieved by a combination of research and a broad-based discussion throughout the software engineering community.

- We believe that role separation — defining different roles, such as programmers, software testers, designers, and systems analysts — is likely to be central to approaches that improve the quality of software systems. Separating roles may be done not only to separate professional from nonprofessional roles, but also to identify professional roles more specialized than the role implied by professional engineering registration. As a rough analogy, it may be that we are better positioned to define the analog of selected board specializations (e.g., software testers) than the analog of the general medical doctor degree (e.g., software engineers).

Overall, our assessment has led us to the conclusion that the SWEBOK effort is geared aggressively towards trying to define an overall level of professional practice for all of software engineering that would implicitly provide assurances to the public. Furthermore, we believe strongly that it will fail to lead to an achievable level of professional practice in a reasonable time and that, in failing, it may lead to a situation in which the public is provided with false assurances of the quality of software systems. The professional societies, including ACM, *must* pursue every possible means towards improving the current state of affairs; at the same time, they *must* refrain from pursuing activities that have a significant chance of reducing the public's understanding of, confidence in, and assurances about key properties of software.