Chapter 2: Before SEEPP, 1968-1994

“In the beginning was the word.”
—*Gospel of St. John*

2.1 NATO, DoD, and SEI

The purpose of this book is to reconstruct from start to finish how SEEPP (the *S*oftware *E*ngineering *E*thics and *P*rofessional *P*ractice Task Force) wrote, revised, and won approval for the Software Engineering Code of Ethics and Professional Practice. Studying SEEPP should give insight into how one profession formed, why it formed, and why it adopted a code of ethics. But, like all who try to tell a story, I face the question: where to start? Before there could be a SEEPP, there was an IEEE-CS ad hoc committee to establish software engineering as a profession. That ad hoc committee itself represented several decades of development, development that will help us understand what SEEPP was and what it was supposed to do—in a way the previous history of the universe does not. This chapter briefly describes that development.

The term "software engineering" is one among several candidates for the name of what remains a new, relatively ill-defined occupation. Among the other candidates are “programmer”, “software designer”, “software developer”, and “software architect”. The earliest of these names seems to be “programmer”. “Software engineer” seems to have come into currency after its use in the title of a 1968 conference sponsored by the Science Committee of the North Atlantic Treaty Organization (NATO).[1] The term then expressed an aspiration (as well as, as the organizers intended, “a provocation”).[2] In 1968, writing computer programs (“software”) was still largely a sideline, art, or individualistic craft, most programmers working in their own way on relatively small programs. Few reported (“documented”) much about their methods or about the structure or the quality of their software. Routine software, relatively small programs similar to earlier work, seemed to be satisfactory (“good enough” for the purpose intended). But too often novel programs, especially if large (involving, say, a thousand “man years”), arrived late, proved unreliable (“bug-ridden”), cost far more than originally budgeted, or failed to do much that was promised.[3] The individualism of “traditional programming” did not fit the new scale. Because engineers generally work in teams on large systems, not as individuals, engineering was supposed to provide an alternative to programming practices then current. “Software engineers” were to work in standard ways as other engineers do, keeping records in the way engineers typically do. Software engineers were to be as good at predicting when they would be done as engineers generally are, to be as good about staying within budget as engineers generally are, and to produce programs as useful and reliable as other engineering products are. Software engineering was to be an order of magnitude better than mere programming.[4]

The organizers of the NATO conference, both developers and users of software, may have overestimated engineers. Almost every engineer has at least one story of a project that fell well behind schedule, was poorly documented, went well over budget, or produced a product the performance of which fell far short of expectations.[5] What was important, though, was not so much how good engineers actually are, as how bad programmers seemed to be. The organizers had a point no one at the NATO conference, or after, disagreed with: *even compared to engineers at their worst*, the programmers of 1968 (however much they achieved) looked pretty bad to each other and to those who increasingly asked them for large programs, programs too large for one or even a few programmers to write.[6]

The new name did not catch on quickly. Almost two decades passed before it found an institutional home. But on January 1, 1985, under the authority of the Department of Defense, the Software Engineering Institute (SEI) began work in Pittsburgh. The reason for the Department's interest in software is easy to understand. After World War II, the Department of Defense had slowly moved from buying simple hardware—tanks, radar, missiles, computing machines, and so on—to buying hardware dependent on software. More and more, what gears, mechanical gauges, electrical switches, and similar devices had done was now being done by software. And much that hardware could not do at all or at a reasonable cost was also being done by software. Software was becoming an ever-larger part of what the Department bought, and ever-more central to what it did. As the Department became more dependent on software, it also became more aware of how unreliable the software was and how often the software was a cause of delays or cost-overruns affecting major weapons systems.

In the early 1980s, the Department of Defense decided to do something about its problems with software. It convened a study group, a panel of experts like others the Department organizes from time to time when it has a problem that seems beyond its own experts. In 1983, the study group recommended that the Department create "a federally funded research and development center" (FFRDC) for software. A FFRDC is a special corporation created by act of Congress. Each is dedicated to work with a unit of the US government. Most such centers have an academic connection. For example, the California Institute of Technology has the Jet Propulsion Laboratory (JLP), a FFRDC that works for the National Aeronautical and Space Agency (NASA). The Massachusetts Institute of Technology has Lincoln Laboratories, a FFRDC working for the Defense Department.

The study group described the new center as a "software engineering institute". That description carried a message. The term "software engineering" meant that the new entity was to be concerned with applications rather than with theory—with cost, reliability, timeliness of delivery, and other practical matters, not with computer science generally. The term "institute" seems to have been chosen over "laboratory" to make a similar point. The new center was to develop standards rather than engage in experiments or develop end products (as labs do)—though it might do experiments or help develop products along the way. The new center was to be more like the National Institute of Standards and Technology (NIST)—which is not a FFRDC—than like JPL or Lincoln Labs.

The study group wanted the institutional connection for the new center to be chosen by competition. Because the businesses capable of managing such a center were likely to be government contractors who would benefit directly from the choice of software standards, the study group recommended that the competition be limited to universities (though partnerships with business were allowed). Some twenty universities or groups of universities bid for the center. Carnegie-Mellon University (CMU) won the competition. So, after a few months of negotiation between the Department of Defense and CMU, SEI opened its doors on the edge of CMU's campus, giving the term "software engineering" both a continuing official endorsement and an organization by which to convert a long-standing aspiration into detailed standards for designing, writing, testing, and maintaining software. SEI was soon an important force within programming, engaged not only in setting standards for software but in developing a curriculum for educating future software engineers. SEI even helped to develop guidelines for a course on the ethics of software engineering.[7]

2.2 New Jersey, Licensing, and Fletcher Buckley

By the early 1990s, hundreds of thousands of software-related workers were called "software engineers", did something called "software engineering", and had sophisticated employers willing to pay them to do it.[8] Some engineering societies had special interest groups devoted to software engineering. Many universities had courses with “software engineering” in the title. There was even a two-volume *Encyclopedia of Software Engineering* about to go to press. Yet, "software engineering" was not an ordinary *engineering* discipline. Few "software engineers" had a degree in engineering, that is, an undergraduate or graduate degree from an engineering program.[9] Many software engineers were graduates of a program in computer science having a single course in "software engineering", a course typically taught by someone with a degree in computer science rather than engineering. But some “software engineers” were self-taught, graduates of the school of hard knocks. Some worked to standards any engineer could be proud of, but many fell far short of that. Anyone, even a 1968-style programmer, could claim to be “a software engineer”. Employers, even colleagues, could not easily, or authoritatively, decide who was, and who was not, entitled to make such a claim.[10]

Surveying the state of software engineering in the early 1990s, Fletcher Buckley (himself a software engineer at General Electric) drew the conclusion that it was time to make software engineering a “profession”. Others had similar ideas about the same time. For example, on June 27, 1991, the lower house of the New Jersey legislature passed Bill 4414 to license “software designers” (though some headlines said the bill was only to “certify” them). The bill’s sponsor defended the bill as a way “to protect consumers from unscrupulous developers.”[11] Though the bill was initiated at the suggestion of a “computer consultant” with a certificate earned in England, it does not seem to have had the support of any professional society or of any large user of software.[12] Indeed, passage in the lower house seems to have generated considerable opposition from the large businesses that hired most of New Jersey’s software designers. The bill died in the upper house early in the 1992, but not without posing a question for software engineers.

Here perhaps is the place to explain some important terms: license, certification, and registration. A “license” is a governmental permission to practice the licensed activity. Generally, the permission is granted upon application if certain qualifications are met (usually, education of a certain sort, certain experience, passing a test, and evidence of a good reputation, or some combination of these). In the US, Professional Engineers (PEs) are licensed in this sense. Only they can sign certain documents, perform certain services, or claim to be PEs. Licensure presupposes prohibiting all but the licensed from engaging in the activity in question.

“Registration” is sometimes a synonym for licensing. But generally it indicates a less stringent form of regulation. Those registered have the right to *claim* a certain title, though anyone can perform the corresponding function. So, for example, if software engineers were registered (in this sense) rather than licensed, only registered software engineers could use “software engineer” to describe themselves (or “software engineering” to describe what they do), but anyone could write software for any purpose, apply for jobs calling for “expertise in writing software”, and even advertise themselves as “software designers”, “software developers”, or “software architects”.

Generally, governments manage registration. “Certification” is registration performed by a non-governmental body (such as the Institute for Certification of Computing Professionals). A non-governmental body cannot control who does what work but it can (relying on trademark) control who can legally claim its certification. Anyone can claim to be a “computing professional”, but only someone appropriately certified can claim (as well) to be a Certified Computing Professional.[13]

The distinction between certification, registration, and licensing sometimes becomes muddled when a government makes (private) certification a condition for registration or licensure. Think, for example, of Certified Public Accountants. In the U.S., state governments actually license CPAs, but a private group, the American Institute of Certified Public Accounts, still provides the test that an accountant must pass to be licensed as a CPA. The test is all that remains of certification.

Buckley lived and worked in New Jersey. For Buckley (as for many who have thought about professions), licensing, registration, or certification seemed an important mark of profession. The brief flap over the licensing of programmers in New Jersey seems to have reminded Buckley of the unsettled state of his profession. There was, first, the simple question of what to call it. Buckley thought of himself as a software engineer. But the New Jersey legislature had thought of the field (or part of it) as consisting of software *designers*—after some engineering societies complained about the use of the term “engineer” for a field they did not consider part of engineering. Some of those who spoke out against licensing described themselves as “artists”, a description that “designer” seemed to authorize; some who spoke out against licensing described themselves as ordinary business people, a description that “developer” might seem to authorize.[14] For Buckley, what was missing from the appeal to art or business was the idea that those who created software, especially complex software upon which life, health, or property depends, should be held to a higher (a more demanding) standard than artists and business people are. It was this higher standard that was central to Buckley’s call to make software engineering a profession. For him, making software engineering a profession was not a way to raise the social status of programmers. In the US, people called “engineers” do not generally have higher status than people called “designers”, “developers”, or even “architects”. Indeed, if anything, the reverse is true. The point of calling the new profession “engineering” and making it a field of that profession was to connect the new field to engineering’s way of carrying on work, a way that engineers generally believe deserves respect (whether it gets it or not). The thinking behind Buckley’s call to make software engineering a profession seems close to that of NATO’s more than two decades before.

Though Buckley may not have been the first to conclude that it was time for software engineering to become a profession, he was especially well placed to do something about it. He had been trained as an engineer. (He had a B.S. from West Point and a M.S. in Electrical Engineering from Stanford.) He had had a long and distinguished association with the Institute of Electrical and Electronic Engineers (IEEE), serving from 1979 through 1990 as a member of the IEEE Standards Board (and of many of the Board’s standing committees). He had an equally long and distinguished association with the IEEE’s Computer Society (IEEE-CS), serving as Chair of the Software Quality Assurance Plans Standards Working Group from 1976 through 1994 and as Chair of the Software Engineering Standards Committee from 1981 through 1983.

In 1984, he became the Computer Society's first Vice President for Standards Activities. He had also served as a member of the IEEE-CS Board of Governors (1984-1986), initiated five IEEE software engineering standards seminars, and received several IEEE awards.[15] Buckley was a software engineer's software engineer.

2.3 Buckley Proposes

On April 15, 1993, Buckley began circulating a draft motion "to establish software engineering as a profession" along with a long justification. Among the venues he chose for publicity was the electronic *Forum for Academic Software Engineering* (FASE)—a long email regularly sent to a list. The May 9 issue contained a letter that began "Dear Educator" and stated that the "following motion or some slightly altered version of it will be discussed in several meetings at ICSE (International Conference on Software Engineering) May 18-21, 1993 [in Baltimore]." The letter also indicted that its author ("we") "will be attending many of these meetings and will have a chance to present information about the motion" and that "there is a good chance that it will ultimately be considered by the IEEE-CS Board of Governors [which was to meet in Baltimore at the same time]." (Since Buckley had just been elected to his second term on the Board of Governors, he could be pretty sure of that "chance" to present the motion.) The letter concluded with an invitation to share "your thoughts" on three subjects:

1. Defining Software Engineering and Software Engineering Ethics.
   Do you think this is possible at this point? Do you consider these definitions exist already (for example, in the SEI materials)?
2. Accrediting Software Engineering programs at universities
   a. Is the Software Engineering curriculum mature enough for accreditation?
   b. Are the universities and professional societies prepared to undertake accreditation in Software Engineering?
   c. What would be the relationship between existing ACM/IEEE-CS accreditation process for Computer Science, the existing ABET accreditation process for Computer Engineering and the proposed ABET Software Engineering accreditation? Can the field and the universities support these subdivisions?
3. Encouraging the states to establish software engineering as a registered engineering field
   Are we ready for this? What would be the impact on your program?

Preceding the letter was a disclaimer from Laurie and John Werth, Vice Chairs for Education of the IEEE Technical Council on Software Engineering (TCSE):

> We have received a couple of replies to our long memo that indicate that the reader believes that Laurie or I authored the motion. In fact the motion is from Fletcher Buckley, a person from outside the ACM Ed Board, the IEEE-CS Educational Activities Board, the SEI or the executive committee of TCSE. In short it is an independent effort, but one with enough steam behind it that the education community needs to think about its response. Let us have your thoughts and your references.[16]

Buckley used the next issue of FASE (May 18) to announce “the motion is scheduled to be presented for approval to the IEEE CS BoG on Friday, 21 May 1993”. Buckley also invited further comments and indicated that he would be making the rounds of the IEEE-CS boards and committees hoping to “gain insight”.

On Friday, May 21, Buckley moved “that the IEEE Computer Society Board of Governors appoint an ad hoc committee to initiate the actions to establish software engineering as a profession.” The motion indicated that the work of the committee should include:

> a. Determining, in coordination with the Standards Activities Board, appropriate definitions and establishing those definitions as approved standards in accordance with IEEE Standards Board policies and procedures.
>
> b. Determining, in coordination with the Educational Activities Board (EAB), the body of knowledge required for a four-year undergraduate curriculum for a Bachelor of Science in Software Engineering and establishing this as an approved curriculum at the Accreditation Board for Engineering Technology (ABET).
>
> c. Determining, in coordination with the Membership Activities Board (MAB), a set of software engineering ethics.
>
> d. Encouraging, in coordination with the MAB and the EAB, states to establish software engineering as a registered engineering field consistent with current practices in civil and electrical engineering. Members on the committee shall be appointed by the chair of the ad hoc committee. Membership on the committee shall be open to all interested parties including non-members of the Board of Governors, the Computer Society, and the IEEE. The committee is authorized to initiate its work immediately.[17]

Buckley’s motion was bound to be controversial for at least four reasons:

First, according to paragraph (b), software engineering is to be a subdivision of engineering (strictly so called), that is, defined by an ABET-accredited baccalaureate program. Though Buckley was himself a long-time member of the Association for Computing Machinery (ACM), he was here proposing something the ACM might oppose. In 1993, software engineers (unlike computer engineers) generally got their degree in software engineering, if any, from a computer science department. Most computer science departments were separate from electrical engineering (and computer engineering) departments. Indeed, many were in the college of arts and sciences (with the other sciences), not in the engineering school. Computer science had its own system for accrediting programs, the Computer Science Accreditation Board (CSAB), still independent of ABET (though cooperating ever more closely with it). To make software engineering a “profession” seemed likely to move the education of software engineers from the computer science department (a science program) to the engineering school (a professional program).[18]

Second, according to paragraph (c), the new profession, even though a new engineering profession, was to have its own code of ethics. For Buckley, apparently, having a code of ethics is central to being a profession. That was a proposition with which most engineers might agree.

But Buckley seemed to assume that the new profession's code would not be the IEEE's or ACM's but something distinct from both. Since both the ACM and the IEEE had recently adopted new codes of ethics, codes with which (presumably) they were still happy, Buckley's assumption that yet another code of ethics was needed seemed to define software engineering as a new profession distinct from both computer science and other parts of engineering, inviting opposition from within IEEE as well as from ACM.

Third, according to the first sentence of paragraph (d), the ad hoc committee was to do all within its power to ensure that software engineers must be "registered". Neither the IEEE nor the ACM had the power to register or license engineers. Their only power was (and remains) to encourage (American) states (the standard jurisdiction for registering or licensing engineers in the US) to register or license software engineers. According to Buckley's motion, the registration was to be done more or less as it was done for "civil and electrical" engineers. This reference to registration of engineers may be intended to assure readers that the sort of licensing in question is not the sort typical of barbers, carpenters, and other non-professionals. But the reference may have had another objective. Of the major divisions in engineering, the electricals had the second lowest rate of registration in 1993 (9%)—with chemicals coming in right behind (8%). Civil engineers had the highest rate of registration (44%).[19] Buckley's mention of electrical engineering in this context was not surprising. Software engineering is closely related to electrical engineering. But the reference to *civil* engineering seems to suggest that the registration should be made as integral to software engineering's status as a profession as it was to civil engineering's (a necessity for any civil engineer in a responsible position on an important project). Most important, this step in the process of making software engineering a profession (encouraging the states to begin registering software engineers) was to be taken immediately—before the ad hoc committee had reported anything back concerning curriculum or code of ethics. The IEEE-CS Board of Governors was, in effect, asked to approve registration before it even knew whether registration was practical.

Asking the IEEE-CS Board of Governors to approve paragraph (d) might also mean preempting a discussion already in progress. Immediately following Buckley's memo in FASE no. 13 (May 18) was a communication on the subject "Licensing and Certification meeting". It simply quoted in full the following (undated) email:

> LAST WEEK I SENT OUT AN ANNOUNCEMENT OF A FORMAL PROPOSAL FOR A STANDARD TO LICENSE SOFTWARE PROFESSIONALS. THIS PROPOSAL WILL BE PRESENTED TO THE IEEE BOARD OF GOV.S AT THE ICSE [International Conference on Software Engineering] CONFERENCE IN BALTIMORE. AN INFORMAL LUNCH ON FRIDAY 21 MAY AT 12.15 IS PLANNED TO DISCUSS LICENSING AND CERTIFICATION. THE LUNCH IS NOT SPONSORED BY ICSE, BUT BY SOME PEOPLE INTERESTED IN LICENSING & CERTIFIC ATION.
>
> WE PLAN TO MEET AT THE NICKEL CITY GRILL (201 EAST PRATT STREET, BALTIMORE 21202, 410 752 0900)
>
> I WILL REPORT TO THE LIST ON THIS AND OTHER DISCUSSIONS ABOUT LICENSING AND CERTIFICATION.

Though this memo says that a licensing proposal would be put before the "IEEE Board of Gov.s at the ICSE", what must have been intended (and understood) was that such a proposal would be going before the IEEE-*CS* Board of Governors. (The IEEE's governing board, the Board of *Directors*, was not meeting at the ICSE.) In fact, no such motion did go before the Board of Governors (except in the subsidiary form of Buckley's item d). Gotterbarn's memo shows independent interest in licensing and—with its invitation to a lunchtime meeting—more opposition to licensing than to other elements of "professionalization". The IEEE-CS Board of Governors had good reason to move cautiously on licensing.[20]

Last, and almost as controversial as the first sentence of (d), are the three other sentences of (d). These, concerning implementation, have no connection with the first sentence—and, indeed, should have been a separate paragraph. They represent the only change Buckley made after beginning to circulate his draft on April 15. According to the three sentences, the ad hoc committee on making software a profession is to be at once wide-open in potential membership and (for an important committee) unusually subject to "stacking". The motion allows anyone in the world to serve on the committee ("any non-member of the…IEEE"). But the chair of the ad hoc committee is to choose the members and may choose whom she wishes without consulting anyone else.

2.4 A substitute motion approved

The details of Buckley's motion are interesting now only in showing the state of his thinking at the time. By a vote of 15-2-5 (15 yes's, 2 no's, and 5 abstain's), the Board of Governors quickly substituted another motion for it. After one further amendment (the addition to paragraph 4 of everything following the bracketed clarification I have added), the following motion was put to a vote:

> That the Board of Governors of the IEEE Computer Society establish an ad hoc committee, appointed by the president, to serve as a steering group for action, planning, evaluation, and coordination related to establishing software engineering as a profession, subject to:
>
> > 1. The ad hoc committee (the steering group) will be composed of well-respected individuals (such as IEEE Fellows from the Computer Society's software engineering community, others of similar standing, and representatives of involved Computer Society units) with a balance of industry, research, and academic backgrounds.
> >
> > 2. The committee will coordinate on this issue with the various Computer Society boards and committees, other professional societies, and other communities.
> >
> > 3. The committee will establish and appoint task forces and working groups as necessary to accomplish the committee's work, with membership open to interested parties including non-members of the Computer Society and the IEEE.

> 4. The various proposals presently submitted are referred to the committee for appropriate action to advance software engineering as a profession. [The committee shall] Consider and document the issues associated with software engineering as a profession, including, but not limited to:
>
> > The factors involved in, and the value thereof, of establishing software engineering as an approved program including the associated accreditation issues.
> >
> > The factors involved in, and value thereof, of establishing a separate set of software engineering ethics.
> >
> > The factors involved in, and the value thereof, of establishing software engineering as a certified or registered field.
>
> 5. The committee is charged to report on its initial plan and program at the time of the November 1993 Board of Governors meeting.[21]

What are we to make of this substitution? One thing is plain. There was little resistance to Buckley's big idea (creating a profession by defining a body of knowledge, an accredited curriculum, a code of ethics, licensing, or some combination of these). Most members of the Board voted for a committee to set in motion a process that could end in software engineering becoming a profession in precisely the sense Buckley intended. What the minutes of the meeting report is not different ways of understanding "professionalization" but different ideas of how to proceed.

The author of the substitution motion was Elliot Chikofsky. Chikofsky was another software engineer (as were an unusually large number of that year's Board of Governors).[22] He had a B.S. in Computer and Communications Science (1972) and an M.S. in Industrial and Operations Engineering (1978), both from the University of Michigan (UM). In the early 1980s, he had been the co-founder and vice-president (chief operating officer) of a UM spin-off, an early developer of computer-aided software engineering (CASE) technology. In 1993, he was director of research and technology for Index Technology Corporation (maker of Excelerator products) and had just become Chair of the Technical Council on Software Engineering (TCSE, at the time, IEEE-CS's largest technical council).[23] Within a few months, the new *Encyclopedia of Software Engineering* would describe Chikofsky as:

> a developer, organizer, and evangelist of software engineering automation…[he] is known for his advocacy of reverse engineering and CASE technologies...a consultant on management issues regarding the application of software technology to solving business problems, as well as on information systems development methods, tools, and design techniques.[24]

On May 18, Chikofsky had chaired a meeting of TCSE's executive committee at which Buckley explained his motion. Chikofsky had invited Buckley because he wanted to see how

Buckley proposed to respond to what Chikofsky thought was a deep split between computer science and engineering on the issue of making software engineering a profession. While Chikofsky agreed that software engineering should be a profession, he did not favor trying to make it one if the attempt would split the software engineering community.[25]

The executive committee had opened the May 18 meeting to anyone who was interested. With between fifty and sixty people present in a relatively small room (assigned the executive committee with a membership half that number), the atmosphere was electric. Though there were a number of questions to Buckley, the meeting soon turned into a series of exchanges between Buckley and Mary Shaw. Shaw was not an engineer. A professor in CMU's Computer Science Department, she had begun teaching there as soon as she received her B.A. in Mathematics from Rice University in 1971 (receiving a Ph.D. in Computer Science from CMU a few years later). She was nonetheless very much a part of software engineering. She had published widely on "value-based software engineering" and taught software engineering at CMU. From 1984 to 1987, she had served as chief scientist at SEI. In 1990, she had published a well-received paper arguing that software engineering was not yet a "true engineering discipline" and could not be until it had developed its "science" a good deal more.[26] At the TCSE's May 18 executive committee meeting, she repeated her published arguments, making clear along the way that engineers and computer scientists might not think alike about the status of software engineering as a profession or an engineering discipline.

Chikofsky soon concluded that he should not support Buckley's motion (as it would be presented to the Board of Governors on May 21) because it settled all the important questions in a way likely to exclude computer science departments from training software engineers. Settling those questions that way would probably create too much opposition from computer science faculty within IEEE-CS (not to mention opposition from the ACM). Buckley's motion would not pass the Board —or, if it did pass, would still not achieve its objective. Opposition would soon make it a dead letter. Chikofsky therefore drafted a substitute in the three days between the TCSE meeting and the Board meeting.

According to the substitute Chikofsky offered, the ad hoc committee would have a general mandate "related to establishing software engineering as a profession" (a mandate so general that few would object). For Chikofsky, what mattered were the procedures (paragraphs 1-5). Apparently, other members of the Board thought otherwise. Chikofsky's motion was soon amended by addition to paragraph 4 of what became its final sentence and a list of "factors" that the ad hoc committee should consider. This amendment was relatively "friendly". Though Chikofsky did not make it, he did second it. It passed 20-1-2. The main motion (as amended) then passed 18-1-4.

As passed, Chikofsky's motion accepted Buckley's conception of profession as including "approved standards", a curriculum, a code of ethics, and licensure (or "registration"), but rejected Buckley's attempt to determine how the categories would be filled in. The final motion allowed the ad hoc committee to fill in the categories as it saw fit, putting off some battles and perhaps avoiding others altogether. The committee might recommend a special accreditation procedure for software engineering programs, ABET accreditation, or none, might recommend establishing a separate code of ethics for software engineering or not, and might recommend licensure for software engineers or not. But, whatever the ad hoc committee recommended, its recommendations were likely to carry considerable weight. The membership of the committee assured that. The committee was to have members both individually "well-respected" and together representing the major constituencies of IEEE-CS, "industry, research, and academic". The committee was to be balanced enough and authoritative enough for its recommendations to have wide support, support on the computer-science side of the IEEE-CS as well as on the engineering side.

2.5 The Steering Committee begins work

Though the May 21 motion set a deadline of November 12 for completion of the ad hoc committee's work, work did not begin immediately. Finding just the right people to serve on the committee took longer than expected. Not until September did the committee "meet" for the first time. There were then only six members (including James Aylor who, as IEEE-CS President, was a member *ex officio*), but each member offered something the others did not. Buckley was one of the six, there no doubt to represent his own ideas. Two others, coming from SEI, would count as researchers. One, Mario Barbacci (chair), held a bachelor's degree from the University of Engineering (Lima, Peru, 1966) and a Ph.D. in computer science from CMU (1973). His specialty in software engineering was quality attributes and software architecture. The other researcher, Larry Druffel, was then in his seventh year as SEI Director. He had a BS in electrical engineering from the University of Illinois, an MSc in computer science from the University of London, and a Ph.D. in computer science from Vanderbilt.[27] Of the remaining two, one, Winston Royce, was, like Buckley, an engineer from industry. Royce's technical work in software engineering at TRW was important enough to earn him a biography in the *Encyclopedia of Software Engineering*.[28] The last member of the steering committee was Stuart Zweben, an academic (professor, Department of Computer and Information Science, Ohio State University).

What explains the structure of this committee? Royce provided industrial weight; Druffel, research weight. Buckley seems to be there because he originated the motion. Buckley would have been the obvious choice for the chair had Chikofsky not actually made the substitute motion ultimately adopted. While Chikofsky was not yet on the committee, he had been asked to serve. With both on the committee, having the committee chaired by one of the other members would prevent any ruffling of feathers. Barbacci was a reasonable choice. Like Buckley and Chikofsky, he favored establishing software engineering as a profession. But, unlike Buckley, Barbacci was not publicly committed. He was then IEEE-CS Secretary; so, he could keep the other members of the Board's Executive Committee informed when Aylor could not attend the ad hoc committee's meetings.[29]

Zweben is more interesting. He was not there as an academic. Norm Schneidewind (professor, Naval Postgraduate School) would soon join the committee to fill that slot. Nor was Zweben there as software engineer (though he was one). Zweben, the only member of the committee without a degree in engineering, was there because he was *ACM* Vice President—and had volunteered. He had volunteered because the preceding month (August) the ACM Council, the ACM's governing body, had adopted the following motion:

> Council endorses the establishment of a Commission on Software Engineering to address the question contained in Council backup, Item 3.7, Pages 3-4 of 7. If possible, this activity shall be done jointly with the IEEE Computer Society. The Executive Committee shall appoint (ACM's) members of the commission, provide it with a reasonable expense

> budget, and take other steps as may be necessary to assure that the commission may complete its work in timely fashion.

Item 3.7 (dated July 15 and bearing Zweben's initials) took the form of a background statement and a proposed action. The background statement observed in part:

> There are ongoing concerns about the persistent inability to construct software systems that are reliable, dependable, usable, on time, and within budget. These concerns are exacerbated by the widening reliance on computers and networks for conduct of all business and other human activities, and by the increasing role of software in supporting human activities of all kinds. The term "software engineering" is typically used to designate the board field of study and practice that addresses these concerns. Terms such as "software design" and "software architecture" also have been used for this purpose, to focus attention on important and fundamental questions that must be answered satisfactorily before good software can be built routinely.[30]

Item 3.7 went on to claim that the ACM had a long history of interest in "these concerns", that a large fraction of its members ("about 40%") were "professionally directly involved with software development", and that almost everything about the professionalization of software engineering should be open for discussion:

> Terms such as "discipline" and "profession" have been used to characterize software engineering, and to argue for varying degrees of regulation for individuals in this field. At the same time, strong objections to these characterizations and to any regulatory efforts have been voiced. There are strong disagreements about whether software engineering is properly regarded as engineering, whether software engineering adequately covers all relevant aspects of design, whether software engineering should be separated from computer science, and whether software engineering is mature enough to be called a discipline….
>
> There is a strong need to clarify terminology, to identify both generally accepted and desirable standards of good software practice, and to further our ability to educate and train individuals to be competent with software engineering and design.[31]

The only action Item 3.7 called for was "[chartering] a Commission on Software Engineering to provide a white paper that assesses the field of software engineering relative to the issues raised above." Then, for good measure, Item 3.7 listed eight specific questions that the commission was to address. Item 3.7 also suggested that the commission should include "leading people from software engineering, academia, industry, and government", should gather "information from other knowledgeable groups and from related published material", and should "cooperate with other organizations having an interest in similar questions". The commission was to complete work by April 30, 1994. In this way, the ACM Council made it possible to cooperate with the IEEE-CS as the ACM "assessed" making software engineering a profession. Clearly, the ACM was less committed than IEEE-CS to making software engineering a profession. The ACM wanted to study the question, not (as IEEE-CS) to take action.

Well before the November 1993 deadline that the IEEE-CS Board of Governors had set for a report on the ad hoc committee's "initial plan and program", Barbacci and Aylor agreed "the magnitude of the task would require the coordinated activities of several task forces, with different assignments and schedules." They therefore transformed the ad hoc committee into a "steering committee" that would define a process and agenda for the working committees and task forces under it (a normal IEEE structure). This transformation was part of what seems to have been a relatively informal process. The committee never met in one place; it conducted business largely by email (then quite new). Only occasionally did a few members of the committee meet for informal discussion when chance brought them face-to-face. Only on November 8 did a majority of the committee (Aylor, Barbacci, Buckley, and Schneidewind) gather in one place (in Santa Clara, California) on the eve of the Board of Governors meeting there.[32]

2.6 The Steering Committee's Four Recommendations

On November 12, 1993, the steering committee (often referred to informally as "the blue ribbon committee") made four recommendations: The first was to adopt "a standard set of definitions". The chief concern here seems to have been the definition of "software engineering" itself. The committee suggested beginning with IEEE Standard 61012:

> Software Engineering: (1) The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software. (2) The study of approaches as in (1).

The task of developing the necessary definitions for the work ahead was to be assigned to the Standards Activities Board of IEEE-CS (and appropriate subcommittees). This recommendation was identical to (a) in Buckley's original motion.

The steering committee's second recommendation was "identification of a required body of knowledge and recommended practices". Recognizing that the knowledge of software engineering was evolving, the committee recommended that the task of identifying the body of knowledge be carried out primarily by "industry experts", those likely to be most up to date. This recommendation was a departure from Buckley's original motion. His recommendation (c) was for the steering committee itself to identify the body of knowledge in cooperation with the IEEE-CS Educational Activities Board. Apparently, the steering committee came to believe that defining the body of knowledge was too big and technical an undertaking for it or a few academics to undertake. Nonetheless, as events were to show, the steering committee drastically *under*estimated the difficulty of defining the body of knowledge.

The steering committee's third recommendation was "to study and customize, if necessary, existing codes [of ethics] already adopted by IEEE, ACM, registration boards, and other relevant organizations." The committee suggested that this task be assigned to the IEEE-CS Committee on Public Policy (COPP). This was another significant departure from Buckley's original motion. His paragraph (d) had anticipated that the Membership Activities Board (MAB) would work out a code of software engineering ethics. COPP now seemed the more appropriate body (though the steering committee did not explain its preference for COPP).

The last of the steering committee's recommendations was that an "academic task force drawn from educational boards within the SEI, ACM, and IEEE Computer Society, and relevant affiliate societies," define curricula for (a) undergraduate, (b) graduate (MS), and (c) continuing education (for retraining and migration).[33] The committee declined to get involved in the potential "turf war" between computer science and engineering about who "owned" the software engineering curricula:

> There is a debate as to whether Software Engineering is a part of Computer Science or vice versa. We should not be distracted by this debate from the goal of meeting the needs of industry. The education needed for competent software engineers could be acquired in different ways. For example, we might identify the need for a foundation on statistics [sic]; at a given school, the courses could be offered by Computer Science, Software Engineering, or other departments. The object is to seek agreement on the curricula that should be taught and not necessarily on which departments teach it.[34]

The steering committee's handling of the curriculum question differs from Buckley's original motion in at least three important ways. First, the committee broadened the discussion to include graduate and continuing education. Buckley's paragraph (c) had concerned only undergraduate education. Second, the committee had broadened those explicitly to be included in the development of curricula (adding SEI and ACM education boards). Buckley had mentioned only the IEEE-CS Educational Activities Board. Third, the committee separated questions of curriculum from questions about the code of ethics. The separation meant that work on the code of ethics could proceed much more quickly and without the political infighting likely to occur when the subject was who should teach what. But that separation also ignored what some thought the logical sequencing of activities: First, define the body of knowledge. Then set standards of practice, including a code of ethics. Only when these were settled would it be possible to decide what should be the curriculum and how accreditation should be done. The development of curricula and code of ethics were to proceed independently of defining the body of knowledge.

The report's "Epilog" indicated that the committee would take up licensing, certification, and other "regulatory instruments" in "our next report". The committee's next report (March 4, 1994) did not do that. Instead, it merely described progress on the four recommendations discussed above. The progress suggests substantial change in how the committee saw its work:

> 1) Define the body of knowledge and recommended practice—we initiated a task force populated mostly with industry people, led by Patricia Douglas of IBM.
>
> 2) Define a code of ethics—we initiated a task force co-chaired by Robert Melford (CS) and Don Gotterbarn (ACM).
>
> 3) Define the curricula—we have agreement from Doris Carver (CS) and John Werth (ACM) to co-chair the task force.[35]

Concerning its first recommendation ("defining terms"), the committee now suggested (after 1-3) that "[this] is probably better done by the three task forces since they are the ones who will identify the terms that need to be in the 'glossary'." The committee also helpfully suggested "various IEEE standards…might provide the bulk of the definitions, so there is not urgent need to have this as a separate 'task force'."[36] In effect, the steering committee had given up this part of Buckley's original plan (along with licensing).

What had happened over the preceding six months to explain these changes in the way the committee saw its work? First, and perhaps least important, the committee's membership had changed. The committee had added Elliot Chikofsky and lost James Aylor (who had completed his term as IEEE-CS president). While Laurel Kaleda had succeeded Aylor as president, she did not join the committee. Zweben remained on the committee and would soon become vice-chair (though only briefly). Second, and more important, the two presidents had worked out a way for the two societies to cooperate on establishing software engineering as a profession. The agreement, presented to the IEEE-CS Board of Governors as a letter from IEEE-CS President Kaleda to Bell, stated:

> The members of the steering committee will include Mario who will continue as chair, a vice-chair (appointed by you as president of the ACM), and six members, three appointed by each society by whatever mechanism and for whatever terms the societies may choose. The steering committee may add additional members (such as the chairs of the task forces established by the steering committee) as ex officio, non-voting members of the steering committee. Mario may identify other details to be worked out, but these are the general guidelines under which we can get started.[37]

The steering committee thus officially became a joint IEEE-CS/ACM committee. The chairing of two of the three tasks forces already reflected the joint nature of the new steering committee. The committee itself now had to reorganize to reflect that joint nature in the same way, becoming the Joint Committee for the Establishment of Software Engineering as a Profession. Out of that reorganization would come SEEPP.

Last, but perhaps equally important, the steering committee's re-defining of its task seemed to arise (in part at least) from increasing clarity about what should, and could, be done, when, and how. We will see more such re-defining of tasks in succeeding chapters.

Notes

---

1. Gary A. Ford and James E. Tomayko, "Education and Curricula in Software Engineering", John J. Marciniak, Editor, *Encyclopedia of Software Engineering*, v. I (John Wiley & Sons: New York, 1994), p. 439.

[2] "The phrase 'software engineering' was deliberately chosen as being provocative, in implying the need for software manufacture to be based on the types of theoretical foundations and practical disciplines, that are traditional in the established branches of engineering." Peter Naur and Brian Randel, editors, S*oftware Engineering: Report on a conference sponsored by the NATO Science Committee, Garmisch, Germany, 7th to 11th October 1968* (Scientific Affairs Division, NATO: Brussels 39, Belgium, January 1969), p. 13.

[3] “The basic problem is that certain classes of systems are placing demands on us which are beyond our capabilities and our theories and methods of design and production at this time. There are many areas where there is no such thing as a crisis—sort routines, payroll applications, for example. It is large systems that are encountering great difficulties. We should not expect the production of such systems to be easy.” *NATO Conference*, p. 16 (quoting K. Kolence).

[4] For a humorous statement of this view, consider the fifth of “Murphy’s Technology Laws”: *If builders built the way programmers program, the first woodpecker that came along would destroy civilization.* Compare Steven McConnell and Leonard Tripp, “Professional Software Engineering: Fact or Fiction”, *IEEE Software*, November/December 1999: 13:

> The most common approach to software development today is code-and-fix programming—hacking. In this approach, a development team begins with a general idea of what they want to build. They might have a formal specification, but probably not. They use whatever combination of informal design, code, debug, and test methodologies suits them. Programmers write a little code and run it to see whether it works. If it doesn’t work, they change it until it does. The code-and-fix approach is far from the state of the art in software development. It costs more, takes longer, and produces lower-quality software than other approaches; its main advantage is that it requires little technical or managerial training.

Or, even more recently, Charles C. Mann, “Why Software is So Bad”, *Technology Review*, July/August 2002: 33-38.

[5] See, for example, Thomas O’Boyle, “Chilling Tale: GE Refrigerator Woes Illustrate the Hazards in Changing a Product—Firm Pushed Development of Compressor Too Fast, Failed to Test Adequately,” *Wall Street Journal*, Monday, May 7, 1990, pp. 1 ff., for the story of a half billion dollar write-off having nothing to do with software.

[6] There is more than one interpretation of this appearance possible. One view, more or less Marxist, is that management was following its usual pattern of trying to turn a skilled craft into a more easily controlled workforce in which “hand work” and “head work” are separate. See, for example, Philip Kraft’s study of “software workers” in the decade immediately following the NATO conference: *Programmers and Managers: The Routinization of Computer Programming in the United States* (Springer-Verlag: New York, 1977). Another interpretation is that this is an attempt to raise the social status of programmers. Counted as “engineers”, programmers will have more respect. A third interpretation, much closer to what participants actually said, is that software was not as good as it should be and that those responsible for it were looking for any way they could to improve it.

[7] For a bit more on these guidelines, and their eventual fate, see Chapter 3.

8. "In the 1991 Computer Society [of the Institute for Electrical and Electronic Engineers] membership survey, over half (54 percent) of the current full members polled indicated that they consider themselves software engineers, as did 40 percent of the affiliate members." Fletcher J. Buckley, "Defining software engineering", *Computer* (August 1993), p. 77.

[9] In the US, an "engineering program" (strictly so called) is one so accredited by the Accreditation Board for Engineering and Technology (ABET, recently renamed "ABET, Inc."). ABET did not then accredit software engineering programs.

[10] Not much has changed since the early 1990s. For details, see John R. Speed. "What Do You Mean I can't Call Myself a Software Engineer?" *IEEE Software,* November/December 1999: 45-50.

[11] Johanna Ambrosio, "Developer Certification Bill Sparks Battle in New Jersey", *Computerworld* 25 (July 15, 1991), pp. 1, 81. In an email to Don Gotterbarn a few days after these events (September 6, 1991), Gary Ford reported: "A campaign by the American Society of Mechanical Engineers was successful in getting the phrase 'software engineer' changed to 'software designer' throughout the bill. ASME seemed not to object to the term 'software engineer' per se, but wanted to reserve it for engineers that satisfied existing requirements (education, experience, passing a test) for licensed engineers." Ford (September 6, 1991) also reported similar legislation being considered (but not passed) in Texas, California, Ohio, and Tennessee.

[12] Steven Levy, "Down by Law: Excuse Me, Sir, But Do You Have a License for that HyperCard Stack?" *MacWorld*, January 1992, pp. 73-82

[13] Gary Ford and Norman E. Gibbs, *A Mature Profession of Software Engineering* (Software Engineering Institute, Carnegie-Mellon University: Pittsburgh, Pennsylvania, January 1996), pp. 12-15. I should add that, *originally*, a "license" was a document someone would be given (such as today's driver's license). Registration had a different administrative apparatus, a list or registry that could be checked if there was doubt about someone's status (such as today's voter registration roll). But this distinction is no longer sharp. Governments keep a record (a registry) of licensed drivers; and many jurisdictions issue a "registration card" (in effect, a license) entitling one to vote.

[14] Mitch Betts, "Industry debates certification", *Computerworld* 28 (May 2, 1994), p.1.

[15] http://standards.ieee.org/reading/ieee/SB/Oct96/obituaries.html. (August 26, 2004) Buckley died in 1996 (at age 63).

[16] I have not been able to track down the "long memo" referred to here.

[17] “Report to the Board of Governors of the IEEE Computer Society from the Steering Committee for the Establishment of Software Engineering as a Profession” (Version: November 15, 1993), Appendix 1, p. 6.

[18] See, for example, David Lorge Parnas, “Software Engineering Programs Are Not Computer Science Programs”, *IEEE Software,* November/December 1999: 19-30.

[19] Ford and Gibbs, *A Mature Profession*, p. 15.

[20] See, for example, an email from Paul Robinson (Tansin A. Darcos and Company of Silver Spring, Maryland), dated September 1, 1993, and sent to Don Gotterbarn’s lists ETHCSE-L@UTKVM1.BITNET (CSEP Archive). Hostility to licensing has a long history in the professions—or, at least, in engineering. See, for example, Sarah K.A. Pfatteicher, “Depending on Character: ASCE Shapes Its First Code of Ethics”, *Journal of Professional Issues in Engineering Education and Practice* 129 (January 2003): 21-31.

[21] “Report” (November 15, 1993) Appendix 1, pp.6-7.

[22] “Chikofsky: Memories” (notes from a phone interview, March 10, 2003), in CSEP Archive.

[23] *Encyclopedia*, v. I, p.95.

[24] *Encyclopedia*, v. I, p.95.

[25] [From memory (of lost email). Confirm (or disconfirm) in Chikofsky interview.].*****

[26] Mary Shaw, “Prospects for an Engineering Discipline of Software”, *IEEE Software* 7 (November 1990): 15-24. Selected runner-up to best article in *IEEE Software* for 1990, it had already been reprinted in Donald J. Reifer (ed), *Software Management*, 4th ed. (IEEE Press: New York, 1993), pp.486-495). It was soon to be reprinted in *Encyclopedia of Software Engineering*, v. II, pp. 930-940.

[27] http://www-2.cs.cmu.edu/~lifetimeline/Druffel.html (August 27, 2004).

[28] *Encyclopedia*, v. II, pp.1106-1107.

[29] Mario Baracci, Email to Davis, 27 January 2003.

[30] “Report” (November 15, 1993), Appendix 2, p.8.

[31] “Report” (November 15, 1993), Appendix 2, p.9.

[32] “Report” (November 15, 1993), pp. 1-2.

[33] The idea seems to be that engineers entering ("migrating" to) the United States (of which, in good times, there are generally a good number) might need re-training of a special sort.

[34] "Report" (November 15, 1993), p. 4. The capitals here are an interesting device. They suggest that there exist departments of software engineering more or less on an equal footing with departments of computer science. At that time, there was not a single Department of Software Engineering anywhere in the world.

[35] "Report to the Board of Governors of the IEEE Computer Society from the Steering Committee for the Establishment of Software Engineering as a Profession" (March 4, 1994), p. 1.

[36] "Report" (March 4, 1994), p. 2.

[37] "Report" (March 4, 1994), p. 2.