Chapter 8: English Spring, Version 2a-2.1

Major actions are rarely decided by more than four people. If you think a larger meeting you're attending is really "hammering out" a decision, you're probably wrong. Either the decision was agreed to by a smaller group before the meeting began, or the outcome of the larger meeting will be modified by a smaller group later when three or four people get together.—"Wolf's Law of Decision-Making"[1]

8.1 A schedule proposed

On February 13, 1997, while still trying to solve glitches in the "comdes.doc" attachment (the annotated Version 2.0 minus Preamble), Gotterbarn sent the Joint Steering Committee a "Proposed Schedule and Task Force Structure". This was not exactly the schedule for 1997 the Committee had requested.[2] It was more elaborate, a plan with a schedule imbedded in it. The Schedule began with "general comments" laying out some assumptions upon which the schedule was based. Following the schedule itself was a budget of almost $6000 to cover a website at Rogerson's center (CCSR), two meetings of SEEPP's executive committee, one "meeting with Chair of task force [presumably, with the Joint Steering Committee]", and "printing costs, page costs, etc., determined by the societies involved". There was also a post-schedule schedule for publication of the code (January 1, 1998-June 1998).

Gotterbarn delivered the Schedule almost exactly a month later than the date Cabrera and Frailey had set. Either planning the year was much harder than Cabrera and Frailey had anticipated or Gotterbarn had developed a schedule of a kind Cabrera and Frailey had not intended. Though Gotterbarn had not circulated the Schedule to the listserv, he had circulated it to what the Schedule itself identified as a committee *to be established*, SEEPP's "Executive Committee" (EC)—Gotterbarn (chair), Miller ("Software Engineering Coordinator"), and Rogerson, ("Technical Management Coordinator").[3] The Executive Committee was to be part of a larger reorganization of SEEPP. The working groups were to be *officially* abolished.[4] In their place was to be an executive committee and a "panel of experts from Industry, Academe, Military, and the Legal Profession": The idea behind this structure was that:

> A balanced code will address the engineering, the management and the academic sides of software engineering. The structure of the task force reflects that balance. Rather than partitioning the leadership along professional association membership lines [the organizing principle of the *Joint* Steering Committee], the executive committee represents three areas that need to be represented in the code: a software engineering coordinator and a technical management coordinator.[5] The previous organization of the SEEPP task force divided the group along lines of ethical issues into separate working groups. This was not productive.[6] The proposed structure of a single panel of experts allows individual experts to contribute where their specialty is involved. The panel includes members from several professional societies including: the ACM, the IEEE-CS, the ICCP, the BCS, IDP and the Deutsche Gesellshaft Fur Informatik. It also includes practicing software engineers and ethics advisors for major, multinational corporations.

> The panel also has an international representation from places such as Egypt, England, Germany, and Scotland.

The point of this part of the Schedule is not obvious. What it asks permission to do had already occurred more or less a month before. Since mid-January Gotterbarn, Rogerson, and Miller had been working together in just the way they would if the Steering Committee declared them SEEPP's executive committee. The "panel of experts" seems to be the membership of the (enlarged) "Professional Competence Task Force".[7] That task force would work no better for being declared a "panel of experts". Gotterbarn already had the power to call Miller, Rogerson, and himself "an Executive Committee"; he could call anyone he recruited to help SEEPP a "member of the panel of experts"; he could even officially fire all the working group chairs and disband their working groups. On January 6, the Steering Committee had expressly given Gotterbarn the power to do all that and more, "whatever changes in membership or structures are deemed necessary to accomplish this task [driving the task force to completion]." Plainly, the Steering Committee was no longer interested in SEEPP's structure but in what SEEPP could deliver. And, if the Committee had interested itself in SEEPP's structure (as the Schedule invited it to do), it might well have asked what a "software engineering coordinator" (Miller) would coordinate that a "technical management coordinator" (Rogerson) could not, or why there was no "academic coordinator", "military coordinator", or "international coordinator" as well.[8] What was there to "coordinate"?

Gotterbarn did not need to invite such questions. Indeed, the Schedule's list of "deliberables" seems likely to have provided the Steering Committee with more than it could deal with at its February meeting (and more than it had asked for). Under February 14 (1997) were four items the Steering Committee had asked for: "a. Schedule for completion of Code by December 1997[;] b. Statement of Code Architecture[;] c. Statement of Relations of Code to other Codes; d. Second draft of Software Engineering Code of Ethics". To this list, Gotterbarn added (as "Additional Products") two documents the Steering Committee had not asked for: the "Statement of Roles of Codes of Ethics" ("Consensus document describing the role of codes of ethics in professional societies and in particular the roles of a software engineering code of ethics; delivered to the chairs of the steering committee") and the "Code of Ethics Working Papers" ("Documentation of all changes and the reasons for these changes made to the first draft of the Code"). The "Additional Product" may have been a response to questions Dennis Frailey had put in a phone conversation (either on his own or on the instructions of the Steering Committee).[9]

Earlier (Chapter 7) we wondered what point there could be to formalizing Gotterbarn's "kitchen cabinet" as an "executive committee" and the remnant of SEEPP volunteers as a "panel of experts". We may now have at least a partial answer. Each of Gotterbarn's "deliverables" comes with a brief description of process. For example, under *a* ("the Schedule"), Gotterbarn reports: "After review by SEEPP executive committee, Delivered to Chairs of the Steering Committee". The same phrasing occurs under *b* (Statement of Architecture). The phrasing under *c* (Statement of Relations of Code to other Codes) is different: "After review by SEEPP panel of experts, delivered as appendix to Code Architecture to Chairs of the steering committee". The phrasing under *d* (the Code of Ethics) implicitly combines these (and perhaps other procedures): "After review by several entities, redrafted and delivered to chairs of steering Committee". The small group that we had so far observed work informally and in great haste now sounds like a vast organization following its own leisurely procedures. That effect was achieved at the cost of a few titles and some careful phrasing (and without any departure from the truth).

Gotterbarn introduced the schedule as an “attempt to meet the Steering Committee’s request that this project be completed by December 1997.” In case the Steering Committee missed the point of “attempt”, Gotterbarn followed with two “general comments”. The schedule is “very tight” *if* “completion by Dec 1997 means completion of the code which reflects a consensus of those concerned.” The schedule is also very tight if the code “ought to [reflect] the results of the other task forces” because that would mean that the satisfactory completion of the code and statements of practice” is tied to “the progress of the other task forces.”

In fact, the schedule is “very tight” even without the *if*’s. The IEEE’s standards-approval process alone could have taken more than a year. It was a process with its own pace. More worrisome than any delay caused by that process was having to wait for the other task forces to complete their work before Gotterbarn could complete his. So far, the progress of the other two task forces had been slower even than SEEPP’s. The curricular task force seemed to be waiting for the body of knowledge task force to complete work before it did anything. (How could one know what to teach if one did not know what was known?)[10] Meanwhile, after a better start, the body of knowledge task force was finding, as SEEPP had, that everything took longer than expected.

Reorganization at IBM had forced reorganization of the body of knowledge task force. Douglas had much less time to devote to the body of knowledge than she had previously. She had recruited Tony Cocchi (computer scientist, IBM’s Watson Research Center) to help her. Since Cocchi was active in ACM (and Douglas was active in IEEE-CS), the body of knowledge task force had acquired the canonical dual IEEE-ACM chairs just as SEEPP lost them.[11] Because Douglas had had less and less time to devote to unpaid work, the body of knowledge task force had taken much longer than expected to develop its survey. By February 1997, it had a small team of experts reviewing the final form. After that, there would be a “prototype” survey using about fifty volunteers, a review of the results, possible further revision of the survey, and then the actual survey. Once the survey was complete, the task force would have to develop the body of knowledge document, itself a large undertaking. (The Defense Department’s Joint Logistics Command was funding much of their work.)[12] Tying SEEPP’s work to the progress of the body of knowledge task force would not mean just a “tight schedule” if completion was to be by December 1997; it would mean postponing completion by one or more years.[13]

Gotterbarn seems to have designed the Schedule to force the Steering Committee to face up to what it was asking when it set December 1997 as the deadline for completing the code. The December deadline might force Gotterbarn to sacrifice *both* the consensus-building the IEEE standards process worked for *and* coordination between SEEPP and the other two task forces. As if to underline the abandonment of the IEEE process, Gotterbarn concluded his general comments by noting that the “schedule below reflects techniques we used to achieve consensus on the ACM code and make clear to the members of the society that the code did represent that consensus.” (Note the “we” and the “ACM”. Gotterbarn is clearly stepping out of the shadow of IEEE-CS procedure, setting the weight of his own experience against that of the IEEE-CS.) Gotterbarn then laid out the “very tight” schedule. The first date (after February 14) was February 28 by which time the *Steering Committee* was to:

Return Code v2 with comments to the task force,
Return comments on any other documents
Indicate commitment to schedule including

- booking space in society publications for publication of draft code and ballot/survey in September Issues
- support of WEB site at CCSR
- support of announcement in society publication of existence of site and invitation to comment
- permission to publicize this schedule to task force

The next few items will be sufficient to give a sense of just how tight the schedule was (and, based on what we have seen till now, how unlikely it was to be followed for long):

1 March

Task Force begins addressing steering committee concerns and continues revision of the code.

Establish a web site at Centre for Computing and Social Responsibility on behalf of the ACM and IEEE-CS for development and dissemination of the Code. CCSR has significant international email lists and it is where I am during the development of the code. The web site could be managed by them and then turned over to the societies at the end of the project.

1 April

Revised Code and working papers to Steering Committee

15 April

Place draft copy of Code on web site for general comment.

1 June

Completion of Code and associated documents for review by Steering Committee. Begin preparation of survey instruments for Code

15 June

Revisit documents in light of Steering Committee comments

30 June

Finish Revisions

8.2 Fairweather and Prior

Though the Schedule seemed to call for an immediate response, the Steering Committee did not respond immediately. Indeed, it would not respond for more than a month (and then not as Gotterbarn had planned). Gotterbarn could have waited impatiently, as he had in earlier years. Instead, he pushed on, simultaneously writing the Preamble and revising the code. He seldom worked out anything alone. Sometimes he would first talk over his ideas with Rogerson. Sometimes, he and Rogerson would sit together at the same computer screen, discussing comments as they came in. Once Gotterbarn thought he understood what was needed, he would prepare a draft and circulate it to Rogerson, Miller, and two others at CCSR whom Gotterbarn had come to respect, Ben Fairweather and Mary Prior.

Ben Fairweather had just received a Ph.D. in Moral Philosophy (University of Wales, College of Cardiff, 1996). CCSR had hired him because Rogerson thought explaining what "social responsibility" meant was in part a philosophical problem. Fairweather had devoted a chapter of his dissertation to a discussion of codes of business ethics.[14] When Gotterbarn arrived in January 1997, Fairweather was working on a paper concerned with codes of computer ethics ("Why an Incomplete Code is Worse than None at All").[15] He had been looking at codes linked to CCSR's website. He was therefore ready to look at another code—and likely to have strong ideas about what a code should contain. Fairweather knew little about software engineering (though he had started college in computer science, switching to philosophy after the first year). One day in January, Gotterbarn gave him a draft of the code and asked for comment. Fairweather did as asked and was from then on part of the drafting process.[16]

Like many in computer science, Mary Prior began her career doing something quite different. Having received a B.A. in English from the University of Lancaster (1975), she worked for a decade as a librarian. As the libraries in which she worked became more and more dependent on computers, she had become more and more involved in information systems. Eventually, she decided she should know more about computer science than she could pick up on her own. She returned to school, taking a Master of Science in Computer Science from the University of Aston in1988. By the time she received her degree, she was no longer interested in working in libraries. She wanted to teach. DeMontfort offered her a university lectureship in the School of Computing. She accepted. By 1997, she was teaching Systems Analysis and Design, Data Modelling and Database Design, and Comparative Systems Development Methodologies. Even so, Prior did not consider herself a software engineer:

> I work on business applications of software. My concern is the use of computers in business organizations. Engineers build software. I analyze the needs of business to determine what kinds of software they need. I might write specifications for software, but I would not build the software. [17]

Prior first heard about SEEPP from Gotterbarn when he arrived in January but she had been thinking about codes of ethics for at least two years. All she needed to participate was a personal invitation.[18]

Though Fairweather and Prior were both at DeMontfort, Gotterbarn did not work with them in the same way. He communicated with Prior—whose office was just one floor above his—almost entirely by email but with Fairweather in a variety of ways. Fairweather liked to have a paper draft of the code to read, and time to think before responding. As Fairweather recalls, he gave his comments orally at "frequent face-to-face meetings with Don and Simon [Rogerson]," but he also responded by email directly to Gotterbarn—sometimes in response to personal emails from him but sometimes in response to emails sent to the SEEPP listserv.[19] As Gotterbarn recalls it, more of Fairweather's comments were in writing and email than oral.[20]

Gotterbarn had then independently developed at DeMontfort a drafting process remarkably similar to the one I had at IIT. At the core were frequent meetings among three people (Burnstein, Weil, and me at IIT, and Fairweather, Gotterbarn, and Rogerson at DeMontfort). Beyond this core were a few others, most quite distant, emailing contributions (El-Kadi, Mechler, and Norman for IIT, and Langford, Miller, and Prior for DeMontfort).[21] Gotterbarn would present a draft, receive comments, revise as he thought warranted, and then present a new draft (with all changes indicated), asking for responses to the changes. Ideally, he would have continued that process until there were no objections (as I had done), but I, who had no real deadline, had begun drafting Version 1 in May 1996, submitting the final draft in October (almost six months later); Gotterbarn's Schedule allowed about half as long for an undertaking of equal scale (with more or less hard deadlines imposed by the publication schedule of various journals). We should not, then, be surprised if his work shows signs of haste (such as a difference in terminology under different rules). What is surprising is how fast Gotterbarn sometimes worked. For example, the day before Gotterbarn sent the listserv the Preamble, Miller emailed his comments (two-and-half pages single-spaced). Most of the changes Miller suggests are small, but some are not. Some of the suggestions were incorporated into the draft sent out the next day.[22]

Though Miller was in the US during most of early 1997, corresponding with Gotterbarn largely by email, he had a part in the process at least as large as Fairweather and Prior. He always corresponded with Gotterbarn directly (and privately). Not all the correspondence has survived, but what has survived shows Gotterbarn and Miller often thinking together about particular questions of drafting. For example, early on February 5, Gotterbarn sent Miller a rough draft of an early version of 2.0 (but which he called "V1.a")—with questions inserted here and there throughout the document. The email began, "Simon and I had a productive morning. What do you think of the following: 1. Structure. Code starts with a preamble—yet to be worked out—probably when finish[ed] with rest of code." Miller responded to this email the same day, inserting a response after each question. For example, in response to this first question, Miller wrote, "As I wrote before, I like the idea of an extensive preamble."[23]

Similar emails survive for February 9, February 10, and March 17. The March 17 email makes a number of suggestions for the Preamble that Gotterbarn was then rushing to complete. Among Miller's suggestions were replacing "enormous" (in the first paragraph) with "significant" because "'enormous' seems a bit dramatic"; deleting "all" from "all three levels" in the third paragraph; replacing "deeper" with "deepest" in the same paragraph ("third and deeper level"); and rewriting the sentences in this paragraph to put them in the active voice. After discussing them with Rogerson, Gotterbarn followed most, but not all, Miller's suggestions.

8.3 Preamble trouble

On March 18, 1997, less than two weeks before the revised code was to go to the Steering Committee, Gotterbarn emailed the listserv the promised preamble (Preamble v 2.0). What was a single paragraph in Version 1 had become eight paragraphs. The Introduction (name changed to "Preamble") was now a little essay, a page and a half long, single spaced. The first paragraph resembles the original. The first sentence is identical except for the addition of "education" and "social affairs" to the list of roles (previously "commerce, industry, government, medicine, entertainment, and ordinary life). The second sentence replaces the original (partial) definition of "software engineer" ("those who design, develop, and test software") with another ("[those] who contribute, by direct participation or by teaching,[24] to the design and development of software systems").[25] The chief change in the third sentence is the addition of "must". Instead of describing software engineers as people who "commit themselves to making the design, development, and testing of software a distinct, beneficial, and respected profession", it now demanded: "[they] must commit themselves to making the design and development of software a distinct, beneficial, and respected profession." Apparently, one can be a software engineer without such a commitment (the "must" of command always implying the alternative as a possibility).[26] The fourth sentence is exactly the same in both versions ("In accordance with that commitment, software engineers shall adhere to the following code of ethics").[27] But the last two sentences of Version 1, the ones that Frailey and Shaw had criticized[28], are gone, their place taken by seven paragraphs each about the same length as the first.

Like their counterpart in the ACM code, these seven paragraphs are neither numbered nor titled, making reference to them awkward. The absence of titles ("keywords") also makes it hard to see the Preamble's structure and hard to find what is there (or, at least, harder than need be). The casual user may not even realize that anything useful waits in what (compared to the numbered, titled, and minutely divided body of the code) must appear an undigested sprawl of prose. Yet, there is much to ponder in these seven paragraphs. The first three of the new paragraphs—the Preamble's second, third, and fourth—together explain the structure of the code. The second (of the eight) explains the relation of keywords to principles. The third paragraph relates the principles to Gotterbarn's "three levels of ethical obligation" (human, professional, and profession-specific). The fourth paragraph explains the content of the specific clauses, introducing what seems to be another three-level distinction (aspire, expect, and demand), attempting to connect it to the other. The attempt does not seem successful. For example, the following is offered as an example of "Level One": "Aspire (to be human)". Since those addressed (software engineers) *are* human, it is hard to see what (literally) there is to *aspire* to.[29] The intended meaning seems to be that software engineers should aspire to do what humans should do (or should aspire to treat themselves neither as having more rights nor fewer duties than humans as such have). That aspiration has troubles of its own. The standards that apply to humans as such (that is, morality) include many (such as "Don't kill", "Don't lie", and "Don't steal") that seem better categorized as "Level Three" ("demand"). Mere aspiration is not the right attitude to take toward basic moral rules. One conclusion to draw from these three paragraphs is that there is something fundamentally wrong with Gotterbarn's theory of code structure.[30]

The next four paragraphs offer advice on how to use the code. The first of these—the fifth of the eight—warns that the code is "not intended to be all inclusive nor is it intended that its individual parts should be used in isolation to justify errors of omission or commission"; it is also "not exhaustive" and not intended to be "read as separating the acceptable from the unacceptable in professional conduct in all practical situations." The code is not a "simple ethical algorithm". The fifth paragraph having explained what the code is not, the sixth explains what the code is, "[an attempt to] influence you to consider broadly who is affected by your work; to examine if we are treating other human beings with due respect;….and to consider if your acts would be considered worth[y] of the ideal software engineer." Software engineers should interpret the code in such a way that they can answer yes to the question, "[Have I acted] in a manner likely to be judged as the most ethical way to act in circumstances by informed, respected, and experienced peers in possession of all the facts"? The seventh and eighth paragraphs explain the utility of the code. The code "provides support for the software engineer who needs to take positive action by documenting the ethical stance of the profession". It also has an educational use, "stating what is required of anyone wishing to join or continue in the software engineering community". In sum, "[because] it expresses the consensus of the profession on ethical issues[,] it can be used as a guide to decision making and as means to educate both the public and aspiring professionals about the professional obligations of software engineers."

Gotterbarn's covering letter announced to SEEPP's listserv that version 2 (without the Preamble) went to the Steering Committee "last month" (presumably, on February 14) along with the schedule. He did not say what the Steering Committee did with either the schedule or the code, but noted that the schedule included "several internal review cycles" before preliminary distribution for wider comment. After revision in response to the preliminary comments, it would be distributed through "the major publications of each society for review [JULY issues] and balloting on each code element." It is therefore "important to get your comments on both the code and the preamble to me as quickly as possible."

This call for comments on the Preamble had an immediate effect. Prinzivalli (the engineer from California who commented on Version 1) responded the same day, March 18, using the listserv. He had three comments (after thanking Gotterbarn for his "leadership and stewardship of this very important software engineering statement on ethics and professionalism"). First, he suggested changing the word order of the last sentence in the fifth paragraph, moving "given the circumstances" from the end of the sentence to the beginning ("Given the circumstances, these situations require the software engineer to use ethical judgment and to act in a manner which is most consistent with the code of ethics"). Second, he wondered about the sixth paragraph's use of "ideal software engineer". Third, Prinzivalli proposed inserting "all" before "software engineers" in the last sentence of the eighth paragraph ("…educate the public and aspiring professionals about the professional obligation of *all* software engineers").

The only other comments Gotterbarn received seem to be from Fairweather. Fairweather had a dozen suggestions. Some seem intended to add precision (for example, adding "and to influence and enable others to do good or cause harm" after "and to cause harm" in the first paragraph. Some were more philosophical (such as the substitution of "rationality" for "humanity" in the third paragraph). And some were purely editorial (for example, also in the third paragraph, changing "for those effected [sic] by their work" to "for those who may be vulnerable to the effects of their work"). Fairweather's comments survived only in a paper copy in Gotterbarn's files. Beside each of Fairweather's suggestions is Gotterbarn's check, X, or question mark. Gotterbarn believes he made these marks after discussing the comments with Rogerson and Fairweather. He may also have received comments on Fairweather's suggestions from Miller.[31]

8.4 The problem of silence

Not only did the world in general (apart from Prinzivalli) fail to respond to Gotterbarn's March 18 email, even the Steering Committee continued its worrisome silence, a silence so worrisome that Gotterbarn emailed the Steering Committee (and the task force listserv) a "Project Management Status Report" three weeks later (April 7) that began, "We need your response."[32] Gotterbarn seems to have hit upon the idea of a status report only a short time before sending it out.[33] Indeed, there exists in Gotterbarn's files an email that begins:

> DEAR EXECUTIVE COMMITTEE:
> This is where you earn your money !!!!

> I need an hour of your time now. Situation is – No Response from the Steering Committee about permission to circulate code for review. Publication deadlines will cause delays until September unless I get them to move before the end of THIS WEEK.

Though undated, this letter's reference to the need for the Steering Committee to "move" *before the end of "this week"* means that it probably could not have been sent before Sunday, April 6, 1997 (the first day of the week), and since Gotterbarn actually emailed the Status Report late on Monday, April 7, the letter must have been sent before then. Later in the letter, Gotterbarn refers to a few things "we need to do…in the NEXT HOUR." Though it sounds like humorous exaggeration, it was, according to Gotterbarn, the literal truth and, more surprising, he felt sure that, if the two other members of the Executive Committee received the email in time, they would read it and promptly do what he asked.[34]

Gotterbarn was asking a great deal of them (Rogerson and Miller). First, he was asking them to review the report itself, noting "sometimes I have a tendency to step on (and crush) toes." Second, he asked the executive committee to read and "word smith" the "introduction to the proposed article (two short paragraphs explaining what the code was and what those reading it in a publication should do with it). Third, Gotterbarn noted that the code "is not QUITE ready to show the steering committee". There were, first of all, changes (indicated in capital letters) made in response to comments on "draft 2" (presumably, version 2a). The Executive Committee should "comment" but "simply Yea or Nea [sic] will do." There is also the question of where to put Section N—and what N should say. Last, "we still need 3.08 plus principles—when to blow the whistle, and social clauses."

A survey of the document appended reveals many more issues to reolve "in an hour". For example, should clause 1.07 ("Ensure realistic estimates of cost, scheduling, personnel, and outcome on any project on which they work or propose to work") be amended by adding "and realistic estimates of the chances of not meeting those estimates"? Should "work to" replace "aspire" in 1.10 and 1.14? Should the following clause be added under Principle 2: "endeavour to produce software that respects the equality of all humanity: in particular recognize that not all people speak English as a first language, that poorly designed software can prevent disabled people from doing things they could otherwise, and that high-technology products tends to take weath from the poor"? Should the following clause be inserted under Principle 3: "Be aware that all technical judgments impact other human beings. Technical judgments have stakeholders other than employers, clients, and users"? In an hour or so, all these issues were settled and Gotterbarn emailed the Steering Committee his Project Status Report.

The Report begins where the Schedule left off. After reminding the Steering Committee of what he had sent them on February 13, Gotterbarn warned that for the "SEEPP Task Force to move forward[,] we need from you, 1) your comments on the code, 2) approval for the establishment of the web site, [and] 3) approval to distribute the code for comment using society journals." Then, as if to provoke some comment on the code, Gotterbarn wrote, "We have not received any comments on the code from you, so can we presume that we are getting closer to the mark with the code and that the explanations of the changes were satisfactory?" There follows two longer paragraphs, one describing the present state of plans to post the code on the IEEE and ACM web sites, noting that the Steering Committee had not yet provided the names of contacts at IEEE or ACM that Gotterbarn had asked for. The other paragraph made a similar point about publications. The deadline for one crucial publication (ACM's SIGCAS) had already passed, but "the editor is willing to accept the document if I can get it in by the end of the week." The deadline for other publications was getting close. Unless Gotterbarn got the Steering Committee's permission to publish the draft code soon, he would miss the July deadline, pushing publication of the code back to September. Gotterbarn said nothing about budget. He had given up on that. It would have been "a distraction". His goal now was to get the draft published. Once published, it would become much harder to stop.[35]

In case the Steering Committee did not appreciate the import of all these details, Gotterbarn makes the point again, taking a whole paragraph that begins, "The project is at a stand still. The time specifically dedicated to this project is being used waiting for approval to move forward." The paragraph flew its red flags in a way not likely to be missed—in part, by making the project sound in worse shape than it was (though portraying accurately the dangers already in view). The project was *not* "at a stand still" *yet*; the proof that it was still moving forward was just below Gotterbarn's "Report": the "Proposed Product for Publication", v 2.0a of the code. The "a" indicated that the code had been amended since Version 2.0 in ways representing a continuation of work rather that the sort of corrections that "2.1" would normally have signaled. The amendments—(almost always) indicated by capital letters— tell us something about how Gotterbarn (consulting with Miller and Rogerson) worked with the suggestions others offered. For example, we know that Prinzivalli made three suggestions concerning the Preamble. While nothing is done about the first (moving "given the situation" to the front of its sentence), Gotterbarn simply accepted the third (inserting "all" before "software engineers"). His response to Prinzivalli's second comment is more complicated. He did not abandon the appeal to virtue ethics implicit in "the ideal software engineer" (as Prinzivalli's comment seemed to have suggested) but tried to clarify the point of the appeal. The crucial passage in the Preamble's sixth paragraph now read: "consider if your acts would be considered worthy of the ideal PROFESSIONAL WORKING AS A software engineer." Gotterbarn was no mere "scribe".[36]

Many similar changes, large and small, are scattered through Version 2.0a. Among the most important is a series suggesting that Gotterbarn thought the process of weakening the code (making it "more aspirational") may have gone too far. The first hint of this is in the Preamble. The last sentence of the sixth paragraph now read: "Since this code represents a consensus of those engaged in the profession[,] one should TAKE INTO ACCOUNT WHAT IS likely to be judged as the most ethical way to act in the circumstances by informed, respected, and experienced peers in possession of all the facts AND ONLY DEPART FROM SUCH A COURSE FOR PROFOUND REASONS, BACKED WITH CAREFUL JUDGEMENT".[37] Though not an algorithm, the code is to be more than a "guideline". Software engineers are not free to interpret it any way they feel is right. They are to take account of what "the profession" would do—operationalized as the probable judgment of one's "informed, respected, and experienced peers". To depart from the code so interpreted, a software engineer must have "profound reasons", nothing less.[38] There will, then, seldom be reason to depart from the code (properly interpreted). In this respect at least, Version 2.0a lays down rules strictly so called (though rules for which there may be a few exceptions). Though the Preamble's fifth paragraph contains new language that may seem designed to weaken the code—the insertion in its last sentence of "spirit of the" between "consistent with the" and "code"—, it is in fact part of a larger revision having the opposite effect. The last sentence of the sixth paragraph provides an interpretation of "in the spirit of the code" (indeed, a relatively demanding principle for interpreting the code). What is in the spirit of the code is generally acting as the most informed, respected, and experienced members of the profession would interpret the code, departing from that standard only for profound reasons.

Among other changes in Version 2.0a that make it more demanding than Version 2.0, some are small changes. For example, clause 1.04 now read, "Ensure proper AND ACHIEVABLE goals and objectives for any project on which they work." It is no longer enough to ensure that the goals are proper (that is, not contrary to technical, legal, or other relevant standards); the goals must also be practical. Clause 1.07 was revised to require a "risk assessment of these estimates [of cost, scheduling, personnel, and outcome]". The revision of 2.01 is a bit more subtle. In Version 2.0, 2.01 required software engineers to "[disclose] to appropriate persons or authorities any actual or potential danger that the software or related documents on which they worked…posed." If they were wrong about a danger, their conduct would be unjustified (though excusable). Under Version 2.0a, software engineers would be justified in "blowing the whistle" if they merely "*reasonably* believe" that such a danger exists.

Other changes are larger, the insertion of a whole new clause (or something very close to that). So, for example, a new 1.12 had been inserted ("Delete, whenever appropriate, outdated or flawed data")—with the clauses following renumbered accordingly.[39] A new 2.05 (under Public) required software engineers to "[endeavor] to produce software that respects diversity". The clause then went on (in a way violating the code's general format) to explain the clause in a full sentence: "Issues of language, different abilities, physical access, mental access, economic advantage, and allocation of resources should all be considered." Principle 5 (formerly N) now read, "A software engineer in a management or leadership capacity SHALL ACT FAIRLY AND SHALL ENABLE AND ENCOURAGE THOSE WHO THEY LEAD TO MEET THEIR OWN AND COLLECTIVE OBLIGATIONS, INCLUDING THOSE UNDER THIS CODE. IN PARTICULAR, THOSE SOFTWARE ENGINEERS IN LEADERSHIP ROLES SHALL AS APPROPRIATE". The old Principle 5 had become Principle 6 (with subsequent principles renumbered accordingly).

Version 2.0a was still plainly a work in progress. So, for example, the switch from "assure" to "ensure" remained incomplete. Principle 1 itself retained "assure", though "assure" no longer appeared in any of the clauses under it. The first two of the nine clauses of Principle 5 also retained "assure" (as they did when they had an N in front of them in Version 2.0). 5.01 read "Assure that employees are informed of standards before being held to them" and 5.02 "Assure employees know the employer's policies and procedures for protecting passwords, files, and other confidential information." "Assure" also appeared in 4.02 ("Assure that any document upon which they rely has been approved by someone authorized to approve it"). Since these clauses were the same as in the previous version, we might suppose that they remained the same because Gotterbarn had not looked closely at them, his attention focused on the clauses he was rewriting. But that explanation will not do. Clause 6.02 retained "assure", though revised to read: "Assure that clients, employers, and supervisors know of the software engineer's commitment to this code of ethics, AND THEIR OWN RESPONSIBILITIES UNDER IT."[40]

One change in the code, one that is nowhere noted and may have been inadvertent, is in the name. Version 1 was titled "The Code of Ethics for Software Engineers". The point of using the noun "Engineers" was to suggest that the code applied to the members of a profession (software *engineers*), not just to anyone performing a certain function (software engineering). Version 2a (April 17, 1997) is titled "Software Engineer*ing* Code of Ethics" (changing the emphasis to function). The only evidence that the change in title was inadvertent is that the short essay introducing the code refers to the code by its former name (as if nothing had changed).[41] Could this (significant) change of title really have happened by accident? Perhaps. (I did not noticee the change until I began writing this book.)

Version 2.0a contained one more innovation worth comment. Unlike preceding versions (but like the ACM's code), it ended with a (still rough) statement of credit:

> This Code was developed by the ACM/IEEE-CS joint task force on Software Engineering Ethics and Professional Practices Task Force: Donald Gotterbarn, Chair; Keith Miller and Simon Rogerson, Executive Committee; Members: Peter Barnes, Steve Barber, esq., C.S. Burstein, Amr El-Kadi, Ben Fairweather, Milton Fulghum, N. Jayaram, Tom Jewett, Maj. A. Kanko, Ernie Kallman, Duncan Langford, Joyce Currie Little, Ed Mechler, Manny Norman, Douglas Phillips, Peter Prinzivalli, Mary Prior, Patrick Sullivan, John Weckert, S. Weisband and Laurie Werth.

This statement of credit is interesting both for who is credited and who is not (or, rather, who is listed as on the task force). "C.S. Burnstein" seems to be IIT's Ilene Burnstein (the "C.S." being her department's initials).[42] She is the only one of IIT's three participants to be listed. She is, however, not the only one of SEEPP/E's members to be listed. El-Kadi, Jayaram, Mechler, Norman, and Sullivan are all there. So, of the IIT-SEEPP/E contingent, only Weil and I are missing (though the third, Burnstein, is wearing false initials). All members of the original SEEPP task force but two are there: Barber, Little, Miller (though for other reasons), Sullivan, Weisband, and Werth. Only Medford and MacFarland are not. We know why Medford is not, but what about MacFarland? Was he omitted by oversight, or because he had resigned? Resignation does not seem to explain MacFarland's omission. Barber had resigned (October 1995) without ever working on the code (though he had done much work on other things earlier). Yet he *is* listed.

Of the remainder of the list, at least two (Fairweather and Prior) had commented on the code but never asked to be on the task force. Four others (Fulghum, Kanko, Langford, and Prinzivalli) had both joined the task force in response to the Call and had commented on one draft or another. Jewett, Kallman, and Phillips had joined the task force in response to the Call but had not commented on the code. Of the several dozen who had joined the task force in its first year and did nothingafterward, they are the only three listed.[43] Peter Barnes (British) seems neither to have joined the task force at any time nor to have had anything to do with the code. His presence on the list is a mystery (not least to him).[44] John Weckert (an Australian), whom Gotterbarn had recently seen at a conference, had (again) expressed an interest in the project but had not yet done anything. [45] Missing from the list—beside Weil and me—is the German "expert" (or, indeed, anyone from the Continent) that Gotterbarn had claimed for the task force ("the panel of experts") less than two months before.

Gotterbarn may have had the same purposes in view in this list as the ACM had when it credited both those who helped draft the code and those who, after the drafting was done, lent the proposed code their authority. Gotterbarn was certainly willing to credit everyone who had helped in any way with the drafting. He had omitted Weil and me only because he thought that, as researchers studying the process, we did not want to have attention directed to our part in the process. When (many months later) we indicated otherwise, he immediately added our names to the list. [46] What seems plain from the list is that Gotterbarn had not yet worked out what should count as a good reason (other than helping with the drafting) to be on the list. Listing members of the task force worked only if those listed were still members of the task force and only if everyone who contributed was a member. But some (like Barber) were no longer members and others (like Prior) never were. Listing only those who actually contributed something to the code would have meant leaving out many who, though not authors of the code strictly speaking, could lend authority crucial to winning the code's adoption (or had aided SEEPP's work in other ways). We may, then, expect this statement of credit to go through at least as many revisions as the code—for much the same reason (the difficulty of finding the right balance between principle and politics).

8.5 The Steering Committee responds

A day after Gotterbarn sent out the "Status Report" (with Version 2.0a appended), he received an email from Frailey (the Steering Committee's vice chair) announcing the Committee's double response. The Steering Committee had met that day (April 8) and agreed to authorize publication of the code "in newsletters, web pages, and other non-archival publications…to solicit inputs from software engineers". There were, however, two "provisos": First, there was to be a "cover memo" from the Steering Committee to "accompany all copies of the draft." Stuart Feldman, an ACM-appointed member of the Steering Committee, was to "get with you on the cover memo, which the Committee must approve". The intent of the cover memo is to make clear what "it [the draft] is and is not." Second, the Steering Committee wanted its "cover memo" to appear along with the code on both the IEEE and ACM web pages (preferably on the IEEE web page with a link to ACM). Barbacci "will get with [Gotterbarn] on the webmaster." [47]

That was the Committee's first response to Gotterbarn's Status Report. It was, in a back-handed way, good news. The Steering Committee had broken its long silence and, in effect, accepted the schedule Gotterbarn had sent them in February and tried to keep to ever since. Gotterbarn's "Status Report" had worked. The Steering Committee had acted with unbelievable speed (meeting, it turned out, by conference call).[48] Frailey's email itself seemed to show that the Committee understood the need for speed (or, at least, feared its message might be lost). The email began, "Don, please acknowledge receipt of this." Unfortunately, that was the only evidence that the Committee appreciated Gotterbarn's worries. The Committee had not done what Gotterbarn had asked (comment on the code, approve establishing a web site, and help with getting the code published in appropriate journals). Instead, it had done precisely what he had asked them not to do; it had introduced "small delays" (the time necessary to get their approval of the cover memo) that could "cause the project to drag out".

The request for the cover memo was the Committee's surprising first response to the Status Report. Even more surprising was the second. The Committee wanted (Frailey said) to "issue certificates of appreciation to those who made significant contributions to either draft of the code." The certificates were to be distributed during the awards ceremony of the (IEEE-ACM-sponsored) 1997 International Conference on Software Engineering (ICSE '97)—to be held in May. Gotterbarn was to prepare the list and send it to Elliot Chikofsky (an IEEE-CS appointed member of the Committee)—with a copy to everyone else. Gotterbarn was to include himself on the list of awardees "of course.:-)" On the surface, the idea of honoring those contributing to the code seemed a good idea. They deserved some honor for all their unpaid work. But the timing was odd. ICSE ’97 was just over a month away. Work on the code would not be over by then. The code would have to go through at least one more revision (two if the February schedule was to be followed). And, of course, nothing guaranteed that work on the code would end in success, that the code would be adopted. The awards seemed premature. Was there a hidden message? Was the message, “Sorry about the cover memo”? Or, “Keep up the good work”? Or just, “Look at what we’re doing”? Whatever the message, Gotterbarn accepted the award from Barbacci a few weeks later at a dinner held as part of an IEEE function in Pittsburgh.[49] Miller received his in the mail a few weeks later.[50] Rogerson received his the following year.[51] No one else who had contributed to “either draft” received any award for their work.

Frailey sent his email on Tuesday, April 8, promising to send the (unapproved) minutes of the meeting later (and did as promised before the end of the working day). Gotterbarn does not seem to have read either email until late April 9 (between 10:00 and 11:00 in the evening). He was at The Hague on Tuesday and Wednesday (along with Rogerson) meeting with the Dutch Office of Technology Assessment and making preliminary arrangements for ETHICOMP 98 conference at Erasmus University.[52]

Gotterbarn read both emails when he returned to Leicester Wednesday evening, but waited till the next day (April 10) to respond. There was much to think about—not least those listed as attending the meeting: Cabrera was there only briefly and, apparently, only read “some thoughts” of the absent Zweben (*ex officio* in virtue of being ACM President) had asked him to give the Committee. Boehm and Shaw (both ACM-appointed members) were also absent. Even meeting by conference call, the Committee had in fact had only six of its ten members present for most of the meeting (only two of them ACM appointees). Was the Committee, or at least the ACM-appointees, losing interest? In spite of the low attendance at the meeting, the Committee agreed to meet “regularly” by conference call, “perhaps every three weeks” (at a time “to be agreed on by committee members”). Had the Steering Committee failed to respond to Gotterbarn over the last two months because it had not been able to arrange a face-to-face meeting? There was evidence of that. Though apparently called in response to Gotterbarn’s Status Report, a third of the meeting (or at least a third of the minutes) was devoted to the “Survey for Body of Knowledge”. The body of knowledge task force had completed its survey (or, rather, a pilot) and was to have certificates just as the SEEPP task force was. “All” members of the Committee were to “respond to the call for dialog [concerning the survey] with well-conceived analysis and recommendations—Deadline will be April 30, 1997.” There was a similar “action” for the code of ethics. “All” were to provide “feedback to Don Gotterbarn on the draft code by no later than May 31.” There was nothing about the curriculum task force.[53] Was it still waiting for the body of knowledge task force to complete its work?

Just before midnight on Wednesday (April 9), Gotterbarn opened two emails from Peter Knoke, one entitled “A sample exercise with your draft SE code” and the other, “More from the Far North Ethics Committee.” Knoke taught computer science at the University of Alaska-Fairbanks. Gotterbarn had seen him often at the annual Conference on Software Engineering Education.[54] Though Knoke had not participated in SEEPP online since his original expression of interest (November 8, 1994), his address had migrated to the listserv; he had been reading the email; and the status report had at last given him a way to participate. Knoke had (according to the first email) given his class of fourteen seniors in computer science “your draft code” (which version he did not say), asking them to apply it to “selected parts of ‘Killer Robot’…[and more] recently…to Bill Gates.”[55] Below was “a sample” of the response, one paper about Bill Gates who “[seems] to be widely disliked up here (not by me of course).” Knoke was grateful to Gotterbarn for “helping me by sending the draft code” and would “now like to give you the feedback you asked for.” How would Gotterbarn like to be fed? “For example, will bits and pieces (like this one) be helpful, or would you prefer that I provide some kind of summarizing paper?” The first option “gets quick data”; the second, would take longer. The second email was another student essay. We do not know what, if anything, Gotterbarn wrote in response. We do know he did not add Knoke’s name to the list of those contributing to the code (perhaps because only his students actually made suggestions).

8.6 Pressing forward

Thursday (April 10), Gotterbarn worked on the “cover memo” the Steering Committee had requested. After consulting with Miller and Rogerson, he decided to expand the existing two-paragraph introduction instead of adding another document. By Friday, he was ready to contact Feldman (who had not yet contacted him). The email, eight pages single-spaced, contained: a covering letter (about half a page); a brief essay entitled “Software Engineering Ethics Code”, by “Don Gotterbarn, Keith Miller, and Simon Rogerson” (about a page long); the “Draft Software Engineering Code of Ethics, Version 2.1 April 1997” (the next seven pages including the “credits”); and last, contact information (including Gotterbarn’s Tennessee mailing address). The tone of Gotterbarn’s covering letter is cool but informal. There are no pleasantries. It begins abruptly after the salutation (“Stu”): “Three items. First the ‘cover memo’.” Gotterbarn then admits that he is “not sure what the steering committee had in mind” but thinks “a cohesive document with an introduction and the code [like the one below] would work better.” What follows that is an explanation of that “cohesive document”:

> I wrote an article about the entire project in a book[,] "The Responsible Software Engineer". The article was based on note[s] supplied by the steering committee when Mario was chair and the article was approved by the steering committee. I have abstracted from that article a statement of the background for my task force. This abstract is the first two paragraphs of the document below. This is followed by a paragraph stating that this is a draft code and soliciting comments. Then comes the code followed by contact information. It would help if we could say where at each web site this will be found. Abstracting from an already published document should make the committee approval process easier.

Gotterbarn did not know why Feldman, a high-ranking computer scientist at Bellcore, was asked to work out the cover memo with him. [56] The most likely reason was that Feldman had suggested a cover memo to accompany the code. Why he should have done that was not clear, but why the Steering Committee approved the suggestion was: the Committee was unwilling to do anything that might seem to endorse Version 2.1.

The second “item” in Gotterbarn’s email seems designed to get Feldman to respond quickly. Gotterbarn admits to not having “heard back from SIGSOFT yet, but the editor of SIGCAS [the quarterly publication of ACM’s Special Interest Group on Computers and Society, Tom Jewett,] has already delayed two weeks and was told by the publisher that the final copy must be fed exed by Monday afternoon [April 14].” [57] If Gotterbarn did not meet that deadline, the draft could not appear in that important journal “until September”, three months later. If Feldman did not approve Gotterbarn’s essay quickly, or propose some acceptable substitute, he would be responsible for delaying the code project several months.

The third “item” suggests that Gotterbarn thought that the Steering Committee had a low opinion of his ability to keep the various versions straight. Why else would the Committee be at such pains to assure that he put the same version of the code in every location? Wouldn’t any competent software engineer distinguish each version of a document (with big integers signaling big changes and decimals, whether alphabetical or numerical, signaling little ones)? Though Gotterbarn might have responded angrily to the Committee’s suggestion, he offered an explanation instead:

> I am a version freak. You will note that this code is listed as version 2.1 rather than 2.a. The differences between this version and what I sent the steering committee earlier in the week is: the spelling in the preamble has been corrected, a possessive apostrophe [w]as removed from the word “opinion’s” because it is a plural not a possessive and the capitalization used to highlight the changes was removed.[58]

This should have satisfied Feldman. Gotterbarn had almost done just what the Steering Committee had asked. He had written the cover memo, though in the form of an essay from his executive committee, not a memo from the Steering Committee. Gotterbarn had suggested that Feldman should find no fault with the first two paragraphs of the essay because, after all, they were only an abstract of an earlier article, one Barbacci (the first chair of the Steering Committee and now an ex officio member because he was IEEE-CS president) had already approved. The other three paragraphs in the essay were, while unconnected with Barbacci, basically true, (more or less) necessary, and pretty much what the Steering Committee seemed to want (a statement of what the draft “is and is not”). The code was a draft distributed “to solicit comments”; the Steering Committee had reviewed the draft; and so on. Yet Feldman did not immediately respond, “Okay, I’ll clear this with the Committee as fast as I can.” The “cover memo” was now a roadblock Gotterbarn had to get around—quickly.

The next day (April 12), a Saturday, Gotterbarn emailed Barbacci a page-long email (with his email to Feldman and the accompanying documents swelling the message to eight pages). He began this email as he had ended the one to Feldman, explaining the differences between 2.1 and “2.0a”, but then went on to ask that Barbacci “only circulate version 2.1”. Since Barbacci had been given the web site as his “action”, that was a reasonable reminder. But it also had the effect of hinting that Gotterbarn had the same concern about the Steering Committee that it had about him (that they would not be careful enough about keeping versions straight). Gotterbarn’s next paragraph took Barbacci beyond his own “action”. It noted that “the steering committee minutes”, as Gotterbarn understood them, said “that the draft code cannot be circulated without an officially sanctioned cover memo.” Would Barbacci look at “the introduction [below] which I believe serves the purpose of a cover memo”? Gotterbarn had, he said, “made this suggestion to Stu” but had not heard anything from him yet. “Is there,” Gotterbarn wondered, “any chance of meeting the Monday deadline for SIGCAS?” He then tried to make clear how important it was that the answer be yes: “According to the minutes of the steering committee, the draft code can only be distributed in an identical form on all media, so we cannot move forward until steering committee approval is received for my suggestions below [the introductory essay].”

Gotterbarn’s next paragraph returned to the subject for which Barbacci actually had responsibility, publication of the code “on the web”. Gotterbarn seemed to have understood the Steering Committee’s plans for publication on the web to include a “chat room” or other arrangement allowing people to respond to one another’s comments.[59] Gotterbarn did not like the idea: “I would rather have responses to the draft code and not have commentators engaging in acrimonious debates with each other.” He feared that the acrimony would discourage many would-be commentators from commenting because “[people] are less reticent to express their views on morals in a private setting.” He then acknowledged that “this is more work for us” (presumably, “this” being the culling of the comments for ideas rather than letting debate settle matters) but, appealing to his own experience, he pointed out that “we managed to handle this kind of email when we did the ACM code of ethics.” Gotterbarn concluded this discussion of web publication by asking what the planned location “at the IEEE site” would be so that he could “be more specific in the article below” about how to submit comments.

Gotterbarn’s email to Barbacci would seem as cool as the email to Feldman were it not for its closing paragraph: “I really appreciate the promptness of your responses. It is fun to work on a project that is moving along. Thanks.” There followed a “ps”: “I will get the addresses, email, and fax information for the executive committee awardees to you shortly.” Why Gotterbarn did not email Chikofsky directly is not clear. Gotterbarn seemed to be treating Barbacci as if he were still the Steering Committee’s chair. In any case, that is the last we hear of the awards in the emails, though Gotterbarn did take time on Friday to straighten out some email addresses. He had tried to make these changes on April 7 (before sending off the status report), working through his George Washington University account. Apparently, that attempt had failed. The very next day he had received a short response to the status report from someone whom he had just removed from the list.[60] He now wrote to the person in charge of listservs at ETSU, asking that seven addresses be removed (including one mistakenly assigned to me) and that four others be added (my correct address, as well as addresses for Barnes, Jewett, and Weckert).[61] SEEPP no longer consisted of everyone who applied (and did not resign).

Whether Barbacci contacted Feldman, we do not know. What we do know is that Feldman responded on Monday (April 14, 1997), about noon, leaving Gotterbarn time to meet the SIGCAS deadline. Feldman began by agreeing that a separate memo was not needed “but” (he goes on to say) “we” (presumably, the Steering Committee) “believe that it is important to explain the approach, tone, and content of [t]he code” (in the essay). In particular, it was important to explain that the code was “modeled on other codes of ethics for recognized professions, that the main purpose of such codes is to educate practitioners about expectations, and to clarify the relations between the professions and the rest of society.” For Feldman, “the legalistic tone” of such codes was a “natural way to describe such a contract” but did not imply that the code will become a “new legal requirement”. The Steering Committee’s worry, or at least Feldman’s, was that the code would be interpreted as a step toward licensure. The Steering Committee (or at least some of its members) wanted to keep the question of licensure out of the discussion.

The remainder of Feldman’s email seems to explain why he took several days to respond. Having stated the Steering Committee’s purpose in asking for the cover memo, he proposed to insert a new paragraph in the essay "after the nice introduction about the purpose and history of the IEEE/ACM collaboration". The paragraph might well have taken a busy researcher two days to compose. It certainly seems designed to help ACM members, especially computer scientists, to accept a code for "software engineers" that does not look much like the ACM's code:

> Codes of ethics are common among professions. The draft below is modeled after several adopted by engineering groups. Although these codes frequently sound legalistic (almost a contract between society and the members of the field), most are not intended as a means of enforcement. Rather, Codes of Ethics are, for the most part, descriptions of ideal behaviors. They instruct practitioners about the standards that society expects them to meet, what their peers strive for and expect of each other. We hope that the following Code will be enlightening as well as defining a standard we plan to meet.

One might suppose that Gotterbarn would have accepted this "friendly amendment" ("friendly" in part because the code's preamble says much the same thing). He did not. When, late in the afternoon, he emailed the SIGCAS Editor the document to be published, it did not contain Feldman's paragraph. Indeed, the introductory essay was exactly as it had been on Friday. Gotterbarn's covering letter suggests both his relief at meeting the deadline and the last-minute lobbying that made doing so possible: "Tom, I finally got it settled. I phoned some people in the US and got clearance!" Gotterbarn no longer recalls what he said in his phone conversation to convince Feldman to abandon his paragraph.[62]

Gotterbarn's files contain an email from Rogerson that seems to explain why Gotterbarn did not accept Feldman's paragraph. The email includes a "snip" from Miller. Preceding the full text of Feldman's paragraph, Miller had written: "I agree [apparently to some comment lost by the snipping]. This makes our code sound even wimpier than it is. I suppose we must bow to the pressures and be somewhat spineless. But we don't have to BRAG about it, for goodness sake." (What seems to have bothered Miller is that Feldman describes the Code as "ideal behavior".) Miller then offered this "reword" of Feldman's paragraph:

> Many professions establish codes of ethics. The draft evolved after extensive study of several engineering codes. All these codes try to educate and inspire the members of the professional group that adopts the code. The code also informs the public about the responsibilities that are important to a profession. Codes instruct practitioners about the standards that society expects them to meet, and what their peers strive for and expect of each other. Codes are not meant to encourage litigation, and they are not legislation; but they do offer practical advice about issues that matter to professionals and their clients AND THEY DO INFORM POLICY MAKERS.[63]

Rogerson told Gotterbarn that he preferred "Keith's version" and certainly would "want to disassociate myself from the previous one." His only suggestion is to add the words in caps to Miller's version.

A week or so after this crisis, Barbacci emailed Gotterbarn to ask for "a list of addresses (Snail and Email) phones, faxes, etc. for all the people mentioned above [in 2.1's credit paragraph]". Gotterbarn responded (April 24) by asking everyone on the listserv to provide the relevant information, adding (pointedly) that "if you want to have your name removed from the list and want to disassociate yourself from the code, please do so now…[and] your name will be removed from the list.” Responses trickled in. The only ones to come through the listserv were for Langford (April 24), Mechler (April 28), and me (April 29).

8.7 Europe responds to Version 2.1

In May, the International Federation for Information Processing (IFIP) met in Corfu. On May 7, its special interest group (SIG) devoted a session to discussion of Version 2.1. The minutes—bearing the bureaucrat’s title “IFIP SIG9.2.2 ‘IFIP Framework on Ethics’”—“record the extensive comments” (a page and a half single-spaced).[64]

After expressing appreciation for the “the effort of the IEEE [sic] task force”, it listed nine “general observations…made by the group”, seven specific “remarks” (organized by Principle), and five “Concluding remarks”. Some of the general observations sound much like the comments Shaw or Frailey made on Version 1: “the code needs to have its items prioritized in a more evident manner. Items need ‘stacking’”; “there are conflicts among some of the items included in the code”; “the code represents idealism; then there is realism, the real world”; “many of the issues raised are not specific to software engineering”; and “the code is really a set of guidance notes”. Other general observations seem to respond specifically to Version 2.1’s innovative Principle 5 (“Management”): “the role of the individual as distinct from the role of the organization should be more clearly delineated”; “would some of the issues be more easily incorporated under a code of business ethics?”; and “are some of the issues raised actually about the management of large projects?” One of the general observations asks about “enforcement” (a question Feldman’s paragraph might have averted). The final observation seems to treat as a vice one of the virtues of the IEEE-CS/ACM effort, its attempt to involve as many people as possible: “some of the names included in the task force appear to be relatively ‘new’ names to the field of ethics”. (The IFIP does not say which names appear new, but certainly the new names must include Burnstein, Mechler, Prinziavalli, and Fulghum, each of whom had done more than some of the well-known names.)

Many of IFIP’s specific remarks took up themes from the general observations. Some are concerned with the “realism” of the code, for example: “The goals [of Principle 1] are too high to be achieved! What is then the meaning of such a statement? Some paragraphs have nothing to do with ethics (1.01 ‘well documented’; 1.05 ‘ensure appropriate methodology’…).” Principle 3 is “Difficult to implement”. Principle 4 requires conduct to be “’consistent with the public health,…’ : does this mean Princ, 4 is ranked over Princ. 3 (see ‘always act…’)?” Principle 7 makes “SIG9.2.2” wonder, “How to interpret 7.03 [‘Credit fully the work of others ‘] when thinking about Microsoft Windows 95 unreliability, or other well known products? § 7.07 [‘Not interfere in the professional career progression of any colleague’] is also a bit of wishful thinking when compared to practice!” Others are concerned with the code’s specificity, for example, “Princ. 7 [‘treat colleagues fairly’]: Being so detailed, why not mention the share of work and responsibility with women. Would it be “gender discrepancy’? But, more precisely, SIG9.2.2 would suggest to insist [sic] on specific software engineering aspects and not on too general matters.”

Other remarks seem independent of the general observations, for example: “In our view, the emphasis [in Principle 2] must be put on ‘disclose the potential danger’. About § 2.06 [‘Be fair and truthful in all statements, particularly public ones, concerning software or related documents'], one could wonder if it is feasible when we know that software always contains errors! We also insist on the difficulty of locating the responsibilities in software development." One remark echoes one of Gotterbarn's own concerns. "Where [in Principle 3] would you include whistle-blowing. In 4.07 ['Inform the client or the employer promptly if, in their opinion, a project is likely to fail, to prove too expensive, to violate intellectual property legislation, in particular copyright, patent, and trademarks, or otherwise be problematic'].?[65] Wouldn't it be better to consider it in a general clause?" The IFIP's remarks on Principle 8 reject Gotterbarn's "relational" understanding of the code: "The title 'Self' is meaningless: SIG9.2.2 thinks 'education' would be more appropriate and needs to be stressed as a principle as such, including the present Princ. 8."

The IFIP's "concluding remarks" seem to be a set of general recommendations. The code should "distinguish the statements that belong to 'Business Ethics in general' and those which are specific to software engineering as such." The "statements [in this code] are individualistic [as are those in many other codes of this kind], whereas software engineering is more and more a collective task, a work of teams where most of the problems are management problems." The code is "too consensual and doesn't recognize the conflicting world in which we evolve." How is the code to recognize conflict? By, for example, saying something "about the problem of delocalization of software engineering [because] it is a real ethical problem of justice." The code would be "better named 'Guidance notes', and surely 'Code of Practice' [would be] better than 'Code of Ethics' (many statements have nothing to see [sic] with ethics; present title is misguiding)." The code should be structured so that "the three levels as mentioned in the Preamble…appear more clearly in the presentation."

IFIP's response to Gotterbarn's Version 2.1 is (despite differences of detail) so similar to the Frailey-Shaw response to Mechler's Version 1 that we must wonder whether the problem lies in a certain conception of codes of ethics rather than in the code itself.[66] That conception seems to rest on at least three dubious assumptions. First, there is some way to settle what should be in a code of professional ethics apart from the practical requirements of the profession. So, the critics want to omit some matters from the code or call the document something other than a "code of ethics". Second, there is a similar presumption concerning the specificity of a code of ethics. Some rules are either too specific or too general for a code of ethics. Third, there seems to be an assumption that one can show that a code of ethics is impractical by pointing out how common is conduct that would violate the code if the code were in force ("wishful thinking"). The criticism at least seems to overlook the possibility that a code can change conduct (as well as overlooking the difference between "documenting" *standards* and "documenting" *conduct*). Insofar as criticism of Version 2 (or Version 1) rests on such a conception of codes, no amount of revision is likely to save the code from the criticism. Those favoring the code must either clear up the conceptual confusions (a philosophical undertaking) or use procedures to convince critics that theirs is a minority view (a political undertaking).

While these IFIP comments probably did not reach Gotterbarn until October, he did receive some comments within a few weeks of the IFIP meeting (though these had no connection with IFIP). The first to arrive were from even farther away. Weckert emailed a half page from Australia on May 14. He thought Principle 1 might need some work: "I'm not sure why usefulness is important, unless it is interpreted very broadly and includes things with no economic value". He suggested "not harmful" in place of "useful".[67] Weckert also suggested replacing "free of error" with "account for error" but admitted that "this is a tricky one." He objected to “reduction of risk in 1.06. Risk should not just be reduced but reduced “to the greatest extent possible”. He wondered whose quality of life 2.02 intended: “In many cases the quality of life of the employer might be enhanced (high profits) while that of employees diminished (loss of jobs or less interesting work). He also wondered whether “client’s” should replace “employer’s” in 4.09 (“Represent no interest adverse to their employer's without the employer's specific consent, unless ethical considerations demand otherwise “). He concluded with a compliment (“what you’ve done is admirable”) and an offer to discuss the comments in person in June should Gotterbarn be at either the International Computer Ethics Conference at Sweden’s Linkoping University or at the Computer Ethics Philosophical Enquiry Conference at Erasmus University. Gotterbarn was to presenta a paper at each.

8.8 American responses

On May 22, Walter Maner, a professor of computer science at (Ohio’s) Bowling Green State University, sent a brief email. After congratulating Gotterbarn on the code’s “organization and content”, he pointed to what he thought was “one large omission”. Those software engineers that do “human factors” research often use human subjects. The code should have a section on human subjects research much like that codes of medical ethics have. There should be something about “informed consent, rights of the human subject, responsibilities of the experimenter, and confidentiality.” Maner volunteered to “take a crack at drafting a set of guidelines”. The offer was never taken up. Gotterbarn does not recall why but suspects it just got lost in the rush of events. Ordinarily, he would have jumped at the chance to work with Maner.[68]

The next day (May 23), another friendly email arrived, this one from the University of Wisconsin at Eau Claire. More than a page-long single spaced, addressed “Dear Mr. Gotterbarn”, and obviously written with considerable care, it was the work of a student, a senior taking Ethics in Computers. His instructor (David Neusse) had emailed everyone in the class a copy of the code and asked each student to provide feedback directly to Gotterbarn.[69] The student had read the code. He was “very impressed by the detail and content”. He thought it “definitely an improvement over the ACM Code of Ethics” which he felt to be sometimes “too general”. He nonetheless had two “complaints”. The first was that Version 2.1 did not stress that “a software engineer should always work to achieve the highest quality possible in his/her work”. In this respect, the ACM code was better. Second, he thought Version 2.1 should have had more to say about the privacy of employee email. The student here seems to have misunderstood what a *professional* code is. It is not a contract with society into which one can insert provisions binding society as well as provisions binding members of the profession. It applies only to members of the profession. Hence, while it can bind managers who are also software engineers, it can bind neither the employer nor the managers who are not software engineers. So, the code can do little about the privacy of email. Though he is a mere student, his idea that a code of professional ethics should bind society as well as the profession has a long history (going back at least to the AMA’s 1847 Code of Ethics).[70]

The student nonetheless concedes that a number of rules under Principle 5 address the issue “somewhat”. 5.01 tells management to “assure that employees are informed of standards before being held to them” and 5.02 that management “assure employees know the employer’s policies and procedures for protecting passwords, files, and other confidential information”. But he would like to the code to state “a more explicit policy towards email in the work place.”

Having made his complaints, he ends his email by listing what he likes “best” in the code. All three items on his list date from Version 1: First, he liked the “last set of principles” because “a software engineer needs to maintain his competence” and “education does not end with a degree”. Second he liked Principle 6, “about a software engineer’s responsibilities to his profession”, especially 6.13 (“Share useful software-related knowledge, inventions, or discoveries with the profession, for example, by presenting papers at professional meetings, by publishing articles in the technical press, and by serving on the profession's standard-setting bodies”) and 6.04 (“Help develop an organizational environment favorable to acting ethically”). Last, he was glad to see “concepts” included in Version 2.1 that were missing from the ACM code, in particular “an emphasis on documentation and testing of software”, “having qualified people work on a project”, “warning the public about the dangers and risks associated with computers”, and “[not asking] employees to do something inconsistent with the Software Engineering Ethics Code”.

Within a week, two more emails arrived from Eau Claire. On May 26, “Student 2” sent a half page of comments. His are much more general than Student 1’s. The introductory essay says that “v 2.1” is based on several other computing and engineering codes. He would like to have the source of each particular provision somehow identified within the code (perhaps as annotations). (No doubt, he would have been happy with something like the table Gotterbarn sent the Steering Committee in February.) Student 2 would also like to know what moral theory the codes in question were based on, for example, “were [they] based on Kantian or Utilitarian principles”? The Preamble mentions three levels of ethical obligation but “there is a deficit of pragmatic examples that could potentially identify and illustrate [the] ethical obligations.” He would like to see some “real life computer related examples” to help the reader “identify with them”. Last, Student 2 wondered whether the list of keywords might be “too extensive in number”. Perhaps some of “the ideas would best be presented in paragraph format with the intent of enticing the readers…[to] process this information.”

The last email to arrive from Eau Claire is even shorter. While “Student 3” liked the code over all, he had two critical comments. First, he thought Principle 1’s statement that software should be “free of error” was unrealistic: “Is there really any large software package that can be ‘free of error’?” Second, he was not clear how the code was to be used. As he read the code, it seemed to say that the software engineer should eliminate all problems at the beginning of the project. He was concerned about what to do if the problems arose thereafter. He knew that the code does not cover “all possible scenarios, but problems are inevitable.” One solution, he suggests, is “to recursively use the code until the project is done.”

8.9 Mechler checks in

Those were the only comments Gotterbarn received on the code in May. The same was true of early June, as Gotterbarn began to pack in preparation for the end of his sabbatical leave and return to Tennessee. Then, on June 19, there was a flurry of emails, beginning with a short email from Mechler at 8:18 AM (EDT) to Gotterbarn’s “computer.org” address: “This is a test from ethics page on IEEE. The address for you was old and I wanted to see if you were getting any comments. Let me know if you get this email.” Twenty minutes later, Mechler emailed the SEEPP listserv: “I do not know if I missed an e-mail reporting the placement of the code on the net. The top web address [at the top of this email] is for the committee which leads to the code and the bottom [also at the top of the email] is the code." An hour later, Mechler emailed Gotterbarn again asking whether Gotterbarn had received the email he sent from the web page (the 8:18 email) or the one he had sent through the listserv twenty minutes later. Mechler was writing because he "could not find a reference on IEEE-CS home page so I search[ed] on ethics. The CS pages are so numerous people may never see the Code. Have you received any comments yet?"

Within a few minutes of receiving this second email, Gotterbarn responded to Mechler's tests: "I received you[r] email. I now have at least three addresses all of which find me where ever I am. Thanks for remindin[g] me about the web site. I will send a message to the group.
He then sent the listserv a status report:

> Sorry to have been slow about this. The Draft code is out for comment in several domains.
>
> > It is on the ACM web site www.acm.org in the serving the community area.
> > It is, as Ed indicated, on the IEEE web site, but somewhat hard to find.
> > It has been published in the June issue of the SIGCAS Newsletter and is coming out in the SIGSOFT newsletter.
> > It should also be out soon in the TCSE-IEEE newsletter. I believe this is the largest IEEE-Computer Society technical group.
>
> All of these presentations ask for comments. There have been no responses yet. We will organize the comments when they start to come in and present them to the task force for discussion and vote. We have not had any new comments from any steering committee member.
>
> Not much else new.

Apparently, Gotterbarn had forgotten the comments received a month ago from Walter Maner and from the three students in Eau Claire. While only Maner's could count as a response to the "presentations", all could count as something "new".

A few minutes after sending out this report, Gotterbarn received another email from Mechler: "I have my son at Vanguard passing the code around and I am sending e-mail to SEs at ERI [where Mechler then worked]. Maybe the task force should send to friends, etc."[71]
Mechler's activity seemed a good omen. Surely, around the world, there were a few thousand, or at least a few hundred, people who, seeing the code in one of the publications, would take the time to respond in much the way the three students at Eau Claire had. The task force might have to sift through a mountain of comments; it would certainly have to do one more draft. But Gotterbarn could now see the end of the project. He was already making arrangements to publish "Version 3.0" with ballots in *Communications of the ACM* and *IEEE Computer* in September or October. He was more or less still on schedule. The project should be over by the end of 1997.[72]

Early in July, when Gotterbarn boarded a plane at Heathrow for the long flight home, he had only a few worries. The first, and most important, was the continued silence of the Steering Committee. Except for the April request for a "cover memo", it had given no direction since it made him SEEPP's sole chair on January 6. It had not interfered, which suggested that it did not disapprove of what he was doing. His version of the code had not had to face the hostile response that Mechler's version had. But the Steering Committee had also not done anything to help, had not formally approved the schedule he had proposed or provided the budget as he had asked. No matter how much he tried, he could not get the Committee to show real commitment. The May 31 deadline for comments from the Steering Committee had passed without anyone on the Committee commenting on anything. He wondered how committed the Committee was to developing a code. He wondered whether he should proceed without express approval of what he was doing. Perhaps he was being too formalistic. When, during a phone conversation, he asked Barbacci whether he should wait for the Steering Committee's approval before proceeding with publication of Version 3, Barbacci had responded something like, "Sometimes you just have to do it. The job gets done and it is easy to apologize."[73] Perhaps Barbacci was right. Gotterbarn had in fact been doing just that since he arrived in England—and look what he had achieved in less than six months!

8. Appendix:

Draft Software Engineering Code of Ethics

Don Gotterbarn, Keith Miller, Simon Rogerson

version 2.1

April 1997

In May of 1993, the Board of Governors of the IEEE-CS established a steering committee for evaluating, planning, and co-ordinating actions related to establishing software engineering as a profession. In that same year the ACM Council endorsed the establishment of a Commission on Software Engineering. By January of 1994, both societies formed a joint committee “To establish the appropriate set(s) of standards for professional practice of Software Engineering upon which industrial decisions, professional certification, and educational curricula can be based.” To accomplish these tasks they made the following recommendations:

1. adopt standard definitions
2. define required body of knowledge and recommended practices
3. define ethical standards
4. define educational curricula for undergraduate, graduate(MS) and continuing education (for retraining and migration).

The steering committee decided to accomplish these tasks through the establishment of a series of task forces. Initially the task forces established were: Software Engineering body of knowledge and recommended practices; Software Engineering ethics and professional practices, and Software Engineering curriculum.

The purpose of the ethics task force is to document the ethical and professional responsibilities and obligations of software engineers. This draft code of ethics was developed by a task force of the Joint IEEE Computer Society and Association for Computing Machinery Steering Committee for the Establishment of Software Engineering as a Profession. The task force on Software Engineering Ethics and Professional Practices developed this code for a sub-specialization within the constituencies of both of the professional societies. In an attempt to reflect the international character of both organizations and the profession itself, the composition of the task force is multinational in both citizenship and in membership in professional computing organizations. The proposed draft Code of Ethics for Software Engineers (version 2.1) was developed by the task force and reviewed by the Steering Committee for distribution and comment as a preliminary draft. The intent of this distribution is to solicit comments from practitioners and other interested parties.

The draft evolved after extensive study of several computing and engineering codes. All these codes try to educate and inspire the members of the professional group that adopts the code. The code also informs the public about the responsibilities that are important to a profession. Codes instruct practitioners about the standards that society expects them to meet, and what their peers strive for and expect of each other. Codes are not meant to encourage litigation, and they are not legislation; but they do offer practical advice about issues that matter to professionals and their clients and they do inform policy makers. Based on the feedback from readers of this publication and from other sources, a final draft of the code will be developed and presented to the Steering Committee for approval.

PREAMBLE v 2.1

Computers now have a central and growing role in commerce, industry, government, medicine, education, entertainment, social affairs, and ordinary life. Those who contribute, by direct participation or by teaching, to the design and development of software systems have significant opportunities both to do good and to cause harm and to influence and enable others to do good or cause harm. To assure, as much as possible, that this power will be used for good, software engineers must commit themselves to making the design and development of software a beneficial, and respected profession. In accordance with that commitment, software engineers shall adhere to the following Code of ethics.

The Code contains eight keyword Principles related to the behavior of and decisions made by professional software engineers, be they practitioners, educators, managers and supervisors, or policy makers, as well as trainees and students of the profession. The Principles identify the various relationships in which individuals, groups, and organizations participate and the primary obligations within these relationships.

Each Principle of this code addresses three levels of ethical obligation owed by professional software engineers in each of these relationships. The first level identified is a set of ethical values which they share with all other human beings by virtue of their humanity. The second level obliges professionals to a higher order of care for those who may be affected by their work. The third and deeper level comprises several obligations which derive directly from elements unique to the professional practice of software engineering. The Clauses of each Principle are illustrations of the various levels of obligation included in that relationship.

The Clauses under each Principle consist of three different types of statement corresponding to each level. Level One: Aspire (to be human); Statements of aspiration provide vision and objectives, are intended to direct professional behavior. These directives require significant ethical judgement. Level Two:

Expect (to be professional); Statements of expectation express the obligations of all professionals and professional attitudes. Again they do not describe the specific behavior details but they clearly indicate professional responsibilities in computing. Level Three: Demand (to use good practices); Statements of demand assert more specific behavioral responsibilities within software engineering which are more closely related to the current state of the art. The range of statements is from the more general aspirational statement to specific measurable requirements.

Although all levels of professional obligation are recognized and because the Code contains different types of statements, the Code is not intended to be all inclusive nor is it intended that its individual parts be used in isolation to justify errors of omissions or commission. The list of Principles and Clauses is not exhaustive, and should not be read as separating the acceptable from the unacceptable in professional conduct in all practical situations. The Code is not a simple ethical algorithm which generates ethical decisions. In some situations standards may conflict with each other or with standards from other sources. These situations require the software engineer to use ethical judgement to act in a manner which is most consistent with the spirit of the code of ethics, given the circumstances.

These ethical tensions can best be answered by thoughtful consideration of fundamental principles, rather than reliance on detailed regulations. These Principles should influence you to consider broadly who is affected by your work; to examine if you and your colleagues are treating other human beings with due respect; to speculate on how the public would view your decision if they were reasonably well informed; to analyze how the least empowered will be affected by your decision; and to consider if your acts would be considered worthy of the ideal professional working as a software engineer. Since this code represents a consensus of those engaged in the profession one should take into account what is likely to be judged as the most ethical way to act in the circumstances by informed, respected, and experienced peers in possession of all the facts and only depart from such a course for profound reasons, backed with careful judgement.

The dynamic and demanding context of software engineering requires a code that is adaptable and relevant to new situations as they occur. But even in this generality, the code provides support for the software engineer who needs to take positive action by documenting the ethical stance of the profession; an ethical foundation to which individuals within teams and the team as a whole can appeal. The code also helps to define those things which are ethically improper to request of an software engineer.

The code has an educational function, stating what is required of anyone wishing to join or continue in the software engineering community. Because it expresses the consensus of the profession on ethical issues it can be used as a guide to decision making and as means to educate both the public and aspiring professionals about the professional obligation of all software engineers.

PRINCIPLES v 2.1

Principle 1: PRODUCT

Software engineers shall, insofar as possible, assure that the software on which they work is useful and of acceptable quality to the public, the employer, the client, and the user, completed on time and at reasonable cost, and free of error. In particular, software engineers shall, as appropriate:

1.01. Ensure that specifications for software on which they work have been well documented, satisfy the user’s requirements, and have the client’s approval.

1.02. Strive to understand fully the specifications for software on which they work.

1.03. Ensure that they are qualified, by an appropriate combination of education and experience, for any project on which they work or propose to work.

1.04. Ensure proper and achievable goals and objectives for any project on which they work.

1.05 Ensure an appropriate methodology for any project on which they work or propose to work.

1.06. Ensure good management for any project on which they work, including effective procedures for promotion of quality and reduction of risk.

1.07. Ensure realistic estimates of cost, scheduling, personnel, and outcome on any project on which they work or propose to work and provide a risk assessment of these estimates.

1.08. Ensure adequate documentation on any project on which they work, including a log of problems discovered and solutions adopted.

1.09. Ensure adequate testing, debugging, and review of software and related documents on which they work.

1.10. Work to develop software and related documents that respect the privacy of those who will be subjected to that software.

1.11 Be careful to use only accurate data derived from legal sources and use only in ways properly authorized.

1.12 Delete, whenever appropriate, outdated or flawed data.

1.13. Work to identify, define and address ethical, economic, cultural, legal, and environmental issues.

1.14. Promote maximum quality and minimum cost to the employer, the client, the user, and the public and make any tradeoffs clear to all parties concerned.

1.15. Work to follow industry standards that are most appropriate for the task at hand, departing from these only when technically justified.

Principle 2: PUBLIC

Software engineers shall, in their professional role, act only in ways consistent with the public safety, health and welfare. In particular, software engineers shall:

2.01. Disclose to appropriate persons or authorities any actual or potential danger that they reasonably believe to be associated with the software or related documents on which they work, or are aware of, may pose to the user, a third party, or the environment.

2.02. Approve software only if they have a well-founded belief that it is safe, meets specifications, has passed appropriate tests, and does not diminish quality of life or harm the environment.

2.03. Affix their signature only to documents prepared under their supervision or within their areas of competence and with which they are in agreement.

2.04. Co-operate in efforts to address matters of grave public concern in software or related documents.

2.05 Endeavor to produce software that respects diversity. Issues of language, different abilities, physical access, mental access, economic advantage, and allocation of resources should all be considered.

2.06. Be fair and truthful in all statements, particularly public ones, concerning software or related documents.

2.07. Not put self-interest, the interest of an employer, the interest of a client, or the interest of the user ahead of the public’s interest.

2.08. Feel free to donate professional skills to good causes.

2.09. Accept full responsibility for their own work.

Principle 3: JUDGMENT

Software engineers shall, insofar as possible and consistent with Principle 2, protect both the independence of their professional judgment and their reputation for such judgment. In particular, software engineers shall, as appropriate:

3.01 Maintain professional objectivity with respect to any software or related documents they are asked to evaluate.

3.02. Affix their signature only to documents prepared under their supervision and within their areas of competence.

3.03. Reject bribery.

3.04. Accept no payback, kickback, or other payment from a third party to a contract, except with the knowledge and consent of all parties to the contract.

3.05. Accept payment from only one party for any particular project, or for services specific to that project, except when the circumstances have been fully disclosed to parties concerned and they have given their informed consent.

3.06. Disclose to all concerned parties those conflicts of interest that cannot reasonably be avoided or escaped and aspire to resolve them.

3.07. Participate in no decision of a governmental or professional body, as a member or advisor, concerned with software, or related documents, in which they, their employer, or their client have a financial interest.

3.08 Temper all technical judgements by the need to support and maintain human values.

Principle 4: CLIENT AND EMPLOYER

Software engineers shall, consistent with the public health, safety, and welfare, always act in professional matters as faithful agents and trustees of their client or employer. In particular, software engineers shall:

4.01. Provide service only in areas of their competence.

4.02. Assure that any document upon which they rely has been approved by someone authorized to approve it.

4.03. Use the property of a client or employer only in ways properly authorized, and with the client's or employer's knowledge and consent.

4.04. Not knowingly use illegally obtained or retained software on equipment of a client or employer or in work performed for a client or employer.

4.05. Keep as confidential information gained in their professional work that is not in the public domain (and is not inconsistent) , where such confidentiality is consistent with matters of public concern.

4.06. Identify, document, and report to the employer or the client any problems or matters of social concern in the software or related documents on which they work or of which they are aware.

4.07. Inform the client or the employer promptly if, in their opinion, a project is likely to fail, to prove too expensive, to violate intellectual property legislation, in particular copyright, patent, and trademarks, or otherwise be problematic.

4.08. Accept no outside work detrimental to the work they perform for their primary employer.

4.09. Represent no interest adverse to their employer's without the employer's specific consent , unless ethical considerations demand otherwise.

Principle 5: MANAGEMENT

A software engineer in a management or leadership capacity Shall act fairly and shall enable and encourage those who they lead to meet their own and collective obligations, including those under this code. In particular, those software engineers in leadership roles shall as appropriate:

5.01. Assure that employees are informed of standards before being held to them.

5.02. Assure employees know the employer's policies and procedures for protecting passwords, files, and other confidential information.

5.03. Assign work only after taking into account appropriate contributions of education and experience.

5.04. Provide for due process in hearing charges of violation of an employer's policy or of this code.

5.05. Develop a fair agreement concerning ownership of any software artifact an employee has contributed to.

5.06. Attract employees only by full and accurate description of the conditions of employment.

5.07. Offer only fair and just remuneration.

5.08. Not unjustly prevent a subordinate from taking a better job for which that subordinate is qualified or experienced to do.

5.09. Not ask an employee to do anything inconsistent with this code.

Principle 6: PROFESSION

Software engineers shall, in all professional matters, advance both the integrity and reputation of their profession as is consistent with public health, safety, and welfare. In particular, software engineers shall, insofar as possible:

6.01. Associate only with reputable businesses and organizations.

6.02. Assure that clients, employers, and supervisors know of the software engineer's commitment to this code of ethics, and their own responsibility under it.

6.03. Support those who similarly do as this code requires.

6.04. Help develop an organizational environment favorable to acting ethically.

6.05. Report anything reasonably believed to be a violation of this code to appropriate authorities.

6.06. Take responsibility for detecting, correcting, and reporting errors in software and associated documents on which they work.

6.07. Only accept remuneration appropriate to professional qualifications or experience.

6.08. Be accurate in stating the characteristics of software on which they work, avoiding not only false claims but claims that might reasonably be supposed to be deceptive, misleading, or doubtful.

6.09. Not promote their own interest at the expense of the profession.

6.10. Obey all laws governing their work, insofar as such obedience is consistent with the public health, safety, and welfare.

6.11. Exercise professional responsibility to society by constructively serving in civic affairs.

6.12. Promote public knowledge of software engineering.

6.13. Share useful software-related knowledge, inventions, or discoveries with the profession, for example, by presenting papers at professional meetings, by publishing articles in the technical press, and by serving on the profession’s standard-setting bodies.

Principle 7: COLLEAGUES

Software engineers shall treat all those with whom they work fairly and take positive steps to support these collegial activities. In particular, software engineers shall, as appropriate:

7.01. Assist colleagues in professional development.

7.02. Review the work of other software engineers, which is not in the public domain, only with their prior knowledge, provided this is consistent with safety.

7.03. Credit fully the work of others.

7.04. Review the work of others in an objective, candid, and properly-documented way.

7.05. Give a fair hearing to the opinion, concern, or complaint of a colleague.

7.06. Assist colleagues in being fully aware of current standard work practices including policies and procedures for protecting passwords, files, security measures in general, and other confidential information.

7.07. Not interfere in the professional career progression of any colleague.

7.08. Not take steps to supplant another software engineer after steps have been taken for employment.

7.09. In situations outside of their own areas of competence, call upon the opinions of other professionals who have competence in that area.

Principle 8: SELF

Software engineers shall, throughout their career, strive to enhance their own ability to practice their profession as it should be practiced. In particular, software engineers shall continually endeavor to:

8.01. Further their knowledge of developments in the design, development, and testing of software and related documents, together with the management of the development process.

8.02. Improve their ability to create safe, reliable, and useful quality software at reasonable cost and within a reasonable time.

8.03. Improve their ability to write accurate, informative, and literate documents in support of software on which they work.

8.04. Improve their understanding of the software and related documents on which they work and of the environment in which they will be used.

8.05. Improve their knowledge of the law governing the software and related documents on which they work.

8.06. Improve their knowledge of this code, its interpretation, and its application to their work.

8.08. Not require or attempt to influence any person to take any action which involves a breach of this code.

8.09. Consider violations of this code inconsistent with being a professional software engineer.

This draft Code was developed by the ACM/IEEE-CS joint task force on Software Engineering Ethics and Professional Practices:

Chair: Donald Gotterbarn

Executive Committee: Keith Miller and Simon Rogerson

Members: Peter Barnes, Steve Barber esq., Ilene. Burnstein, Amr El-Kadi, N.Ben Fairweather, Milton Fulghum, N. Jayaram, Tom Jewett, Maj. Mark Kanko, Ernie Kallman, Duncan Langford, Joyce Currie Little, Ed Mechler, Manuel J. Norman, Douglas Phillips, Peter Ron Prinzivalli, Patrick Sullivan, John Weckert, S.Weisband, and Laurie Honour Werth

Please send comments on this draft of the Preamble and the Code to:

Donald Gotterbarn
Computer and Information Sciences
East Tennessee State University
Box 70711
Johnson City, TN 37614-0711
d.gotterbarn@computer.org

NOTES

[1] Wolf's law seems to be a special case of the "Iron Law of Oligarchy", so named by Robert Michels, *A Sociological Study of the Oligarchical Tendencies of Modern Democracy* (Dover: New York, 1962), originally published in 1915.

[2] Steering Committee\Schedule\SCHEDv2. "Gotterbarn\History of SE Code\History expanded" informs us that the document was sent to the Steering Committee. The Steering Committee minutes for February 1997 are silent about its reception (and also unusually brief about other matters).

[3] Earlier versions of the Schedule go back to a February 2, 1997 draft for circulation within the executive committee: Gotterbarn\Version 1\Schedule vI (which includes some jokes). Miller's email of February 10 concluded with a comment on one of those earlier versions: the "schedule seems fine" but "[you] know the politics, I don't. Unpaid work for all of us, and it seems like lots of it (especially for you), but do what you have to do, and tell me what to do when."

[4] In comments on this chapter (then numbered "7"), Gotterbarn cautioned: "The description you cite about experts was an oversimplified way to describe what I did as reorganization which was sent to the Steering committee. Think of the difficulties you have had with the events I described in 1-4 [now 2-5] above and it will be clear why I did not try to explain stuff about the list serve to the steering committee." Gotterbarn Chapter7cmt (September 30, 2004).

[5] For some reason, the "three areas" have become two. Perhaps Gotterbarn himself represents the third ("academic coordinator"?).

[6] That is hardly an overstatement. For example, Barber recalls: "I was assigned to a working group for one of the principles, but I don't believe that the working group ever met by the time I told Don that I wouldn't be actively participating in SEEPP any more." Interview of Barber, November 14, 2002, answer to question 8. Note that Barber seems to believe that there is some connection between the code's "Principles" and the original working groups.

[7] The list of experts by nationality is a bit odd. On the one hand, who (but a Scot like Rogerson) would count Scotland a country (or nation) separate from England? What happened to the "United Kingdom"? On the other hand, Gotterbarn's early lists of applicants did include several from the Irish Republic. Why no mention of these? There are other oddities as well. For example, none of Gotterbarn's lists ever included anyone from Germany (though a Swiss eventually does comment). Had he made some new contacts (who, though seemingly promising then, did not fulfill their promise)? In comments on this chapter, Gotterbarn described the panel of experts as the Professional Practices Task Force "enlarged by addition of working group leaders—with the exception of Melford—who should have been familiar with what was going on." Gotterbarn Chapter7cmt (September 30, 2004) The nationalities listed (and omitted) suggest a complex (or haphazard) enlargement.

[8] Gotterbarn would have had particular trouble with this last question. Miller does not seem to have coordinated anything. In fact, he seems to have been pretty distant from the administrative side of all SEEPP's work. His February 10, 1997 memo specifically declared an indifference to politics (in an exchange on Principle 1): "If it [the substitution of 'insofar as possible' for 'significant'] is politically required, leave it. I still don't like it, and you know why, but the politics are unknown to me, and I'd like to keep it that way." Another example of Miller's distance from SEEPP is a March 18, 1997 email to Gotterbarn:

> When I was at the APPE conference in DC, I talked to Vivian Weil of the Illinois Institute of Technology. She has been interested in our long and arduous adventure trying to bang out an ethics code. I told her in general about some of our difficulties, but I felt uncomfortable about going into any specifics since I don't know how much of what we have discussed and emailed is confidential…. Anyway, would you please email Vivian and give her a thumbnail sketch of what's been going on? Also, if you think it is appropriate, add her to the list of people getting copies of our code drafts. She is: weil@charlie.cns.iit.edu.

Apparently, Miller did not know (or had forgotten) who was on Gotterbarn's "panel of experts" or how much trouble Gotterbarn had had with that particular email address (and its kin). He also seems to have forgotten that Weil had been a member of his own working group and so someone for whom Gotterbarn (as SEEPP chair) should already have had an address.

[9] Gotterbarn Chapter7cmt (September 30, 2004).

[10] In fact, what was soon known as the "Software Engineering Education Project" (SWEEP) would complete work on accreditation criteria in November 1998. For details, see Gerald L. Engel, "Program Criteria for Software Engineering Accreditation Programs", *IEEE Software* November/December 1999: 31- 34.

[11] Though Cocchi's work at IBM concerned Java (and similar programming) languages, he was trained as an engineer: B. EE. (Pratt Institute), 1964, and M. EE (Pratt Institute), 1967. Two decades later, he did receive a master's degree in computer science from New York Polytechnic, but still considered himself an engineer—"unless you use my distinction between engineer and scientist [as I do:] An engineer delivers things on time, on budget, and on function. Scientists do experimental work. I do experimental work." Interview of Cocchi, November 12, 2002.

[12] Minutes of Steering Committee (February 1997). Douglas and Cocchi submitted their report on the "Pilot Survey" on March 27, 1997, an imposing document of more than a hundred pages.

[13] On April 18, 2001, the IEEE-CS published a trial version of the Guide to the Software Engineering Body of Knowledge (www.swebok.org). For details of development through 1999, see Bourque et al., "The Guide of the Software Engineering Body of Knowledge", *IEEE Software,* November/December 1999: 35-50.

[14] The dissertation's title was: *The Relation of Green Moral and Political Theory to Liberal Moral and Political Theory*.

[15] Eventually published under that title in G. Collste, ed., *Ethics in Information Technology* (Linköping University Press, 2000), pp. 259-277; and reprinted in Terrance Ward Bynum and Simon Rogerson, *Computer Ethics and Professional Responsibility* (Oxford: Blackwell, 2003), pp.142-155. Though this is how Fairweather remembers the circumstances when Gotterbarn arrived (Interview with Fairweather, February 25, 2003), Gotterbarn remembers it a bit differently: "Check this fact: Last time I was at CCSR website—the site had a list of computer Codes—Australian Code etc, which I brought with me to CCSR, Ben had not been looking at these prior to my arrival." Gotterbarn Chapter7cmt (September 30, 2004). Unfortunately, there seems to be no way to reconstruct what was on the site when Gotterbarn arrived. We may, I think, safely assume that some of the codes now present, such as NSPE code, were already there (and that they would have been enough for Fairweather's purposes at that time).

[16] Interview with Fairweather, February 25, 2003.

[17] Interview with Prior, February 24, 2003.

[18] Interview with Prior, February 24, 2003.

[19] Interview with Fairweather, February 25, 2003.

[20] Gotterbarn wonders whether there is a simple explanation of this disagreement: Fairweather, he thinks, may have communicated with Rogerson orally more than with him. Comments on (what was then) Chapter 7 (September 22, 2003).

[21] Langford declined to be interviewed and gave the impression that he had not done much. "Much" is, of course, a relative term. We may gauge Langford's participation from just one example, his page-long email commenting on Version 2.0 that arrived on February 28, 1997. His suggestions were: revising the Preamble to read "free from KNOWN error"; replacing "aspire" in 1.10 and 1.14 with "work" ("aspire" being too "weak"); deleting "authorities" in 2.01 because "they would automatically be the 'appropriate persons'; deleting "and " after "meets specifications, and" in 2.02; replacing "should" with "must" in Principle 3; adding a reference to "law" in the Preamble's "consistent with the public, health, safety, and welfare"; revising 6.02 by adding "prior" before knowledge; and adding "and appropriately" after "credit fully" in 6.03. Gotterbarn accepted many of these.

[22] Email, March 17, 1997.

[23] Gotterbarn credits Miller with being "responsible for much of the excellent rhetoric of the preamble. September 22, 2003 comments on Chapter 7.

[24] Do the drafters really mean that one can be a software engineer without designing or developing software but simply "by teaching" some course to software engineers (Java or even philosophy)? What is it to "contribute" to the design or development of software *by teaching*? Has the "academic lobby" gone too far here?

[25] What happened to "test"? Don't software engineers test software? Or was testing simply assumed to be part of designing and developing? Explaining this change would have been helpful. After all, testing is such an important part of operation and maintenance as well as design and development that it often receives separate mention in descriptions of software engineering, for example, "[software engineers] specialize in designing, building, testing, and 'maintaining' software products." David Lorge Parnas, "Software Engineering Programs Are Not Computer Science Programs", *IEEE Software*, November/December 2001: 19-30, at p. 20. Gotterbarn's definition is almost as distant from the official definition of "software engineering" quoted in Chapter 2.6: "The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software.") That definition, while omitting "testing", does include (as Gotterbarn's does not) "operation" and "maintenance"? We may expect the Preamble's definition of software engineering to develop in succeeding versions. Lists like this are politically important. They recognize groups, especially substantial ones that that want to be remembered.

[26] In comments on the first draft of this chapter (September 22, 2003), Gotterbarn offered a different reading of this "must" as something like the logical "if and only if" rather than a command: "You ain't a professional software engineer without this commitment." I agree with Gotterbarn's claim about what is necessary to be a professional software engineer, but I don't think his use of "must" here makes that point. My ear and his seem to differ (though we are both native speakers of American English), a reason to find another way to express the point we both agree on.

[27] The switch from "must" in the third sentence to "shall" in the fourth invites the question whether the drafters intend "must" to mean the same as "shall" (as it generally does) or something different. That is a question more care in drafting could have avoided. Plainly, we are still dealing with a "first draft".

[28] The two sentences were: "The seven main paragraphs state general rules. Each subsidiary clause is a specific application of its general rule, one experience has shown needs express statement; but no set of subsidiary clauses exhausts the general rule."

[29] Fairweather (apparently seeing the problem noted here) suggested replacing "Aspire (to be human)" with "Aspire (as a human"). Gotterbarn rejected this suggestion and a related one, replacing "Expect (to be professional)" with "Expect (as a professional)". Apparently, he did not see the problem—or at least did not see the change proposed as a solution to it.

[30] Gotterbarn: "[I] later discover[ed] that emphasis on three levels of ethics caused problems because everyone wanted the code clauses to be assigned to a specific level." Gotterbarn\History

of SE Code\Expanded History of SE Code. While the ACM code also distinguishes between several levels, its analysis of those levels is different from Aspire, Expect, and Demand. Its Preamble states: "[This code] contains many, but not all, issues professionals are likely to face. Section 1 outlines fundamental ethical considerations, while Section 2 addresses additional, more specific considerations of professional conduct. Statements in Section 3 pertain more specifically to individuals who have a leadership role, whether in the workplace or in a volunteer capacity, for example with organizations such as ACM. Principles involving compliance with this Code are given in Section 4."

[31] Comments on Chapter 7, September 22, 2003.

[32] On the same day, Gotterbarn sent me the form letter beginning "You have been added to the PRFCMP-L mailing list" with the correct address csep@charlie.cns.iit.edu.

[33] According to Gotterbarn, the status report is a routine part of managing a software engineering project. The purpose of the report is to identify and address problems. But a "status report can only be issued when you have a project plan to report on." The project plan (the one that the Steering Committee had never approved or rejected) was originally developed to accomplish three purposes: "1. it helps keep the project organized. 2. it enables the identification of who—where there are failures. 3. it sets a schedule—so if you (the steering committee) are serious about meeting a schedule then here is what you must do, if the project fails and—as indicated in the status report—you did not do your job, then there is no question as to who is responsible." Comments on Chapter 7, September 22, 2003. Though Gotterbarn had been giving the Steering Committee status reports all along, he had never before given one without being asked to do so. In this respect, this status report is not routine. It is also not routine in reporting on a plan that remained unapproved. Until then, Gotterbarn had always reported on progress in carrying out a plan the Steering Committee had approved. Gotterbarn was, *it seems to me*, in part using the Status Report to push the Steering Committee—and, if they did not do as he wanted, to be sure the blame would be theirs, not his. I stress "it seems to me" because Gotterbarn does not agree with this reconstruction of his motivation.

[34] Comments on Chapter 7, September 22, 2003.

[35] Comments on Chapter 7, September 22, 2003. Note that Gotterbarn also seems to have given up on getting the Schedule approved but nonetheless is not only trying to keep to it himself but (at last) has found a way to tell SEEPP exactly what it says—by the perfectly standard act of copying SEEPP when he sent the report to the Steering Committee.

[36] A comparison of Version 2.0a's Preamble with Fairweather's proposals for revising the first draft, reveals a similar pattern. Gotterbarn accepted some suggestions (those he had checked) and rejected others (those beside which he had put an x or question mark).

[37] The spelling of "judgement" ("e" after the "g") suggests a British origin for the language in CAPS. And, indeed, the language in CAPS appears among Fairweather's suggestions.

[38] What “with careful judgement” adds to “profound reasons” is unclear. The suggestion seems to be that the software engineer must not only have good reasons for departure from the consensus (a substantive criterion) but must also have considered the departure carefully (a procedural criterion).

[39] The “as appropriate” is, of course, redundant, since all clauses under Principle 1 are preceded by “as appropriate”. Redundancy, though inelegant, is not therefore unwise. Sentences in codes of ethics are often quoted out without consulting context.

[40] In comments on Chapter 7, September 22, 2003, Gotterbarn offers another explanation: “The assure-insure debate went principle by principle, some claiming one or the other was stronger, some claiming one or the other involved legal obligations so those principles which would be legally binding got the appropriate label.” This explanation, though plausible in the abstract, does not seem to have any basis in any document surviving from the period. Since “ensure” eventually replaced “assure” everywhere, and transition seems to have begun after Langford suggested just such a complete substitution, the mix in Version 3 seems more likely to be the result of hasty drafting than of debate principle by principle, clause by clause, leaving no trace in the record.

[41] This inconsistency between title and text continues in the publication of Version 3. See, for example, *Computer*, November 1997: 88.

[42] The reason we must go on the evidence is that Gotterbarn no longer recalls. The only other possible “Burnstein” is “Lawrence Bernstein” (at AT&T’s Bell Labs) whose initials are no closer to “C.S.” than Ilene’s and the spelling of whose last name is more distant. But this is not certain, since Gotterbarn seems to have confused them now and then. In a February 10, 1997 email updating addresses, he assigns “Lawrence Burnstein” (sic) the IIT email address: CSBURNSTEIN@HARPO.CNS.IIT.EDU. By then, the listserv had only thirty-four names, and Gotterbarn’s handwritten notations show an intention to drop at least six of those.

[43] Gotterbarn believes that both Jewett and Phillips did email comments as Version 3 was being prepared, but the emails do not seem to have survived either in Gotterbarn’s records or theirs.

[44] Email (Barnes to Davis), January 13, 2003: “I have no recollection of ever being involved in any way in this process. In view of the fact you incorrectly mailed Survive Ltd as “Survival” I wonder if there may be another Peter Barnes in such an organization out there somewhere? Alternatively it may be that I spoke to someone about the development of a code of practice for business continuity and disaster recovery planning professionals while I was at Survive. If so, I have no specific recollection and doubt that I would be able to contribute in any way to your current research.” Gotterbarn (who provided the email address) thinks this is the right Peter Barnes: “[He] may not remember his involvement. He asked to be involved early on via email—his son responded saying Peter Barnes would be interested. We put Peter Barnes on the list. As I

recall, I don't think Peter Barnes ever responded to any further emails about the Code. Five years ago…he may not remember any of it." Whatever Barnes's contribution, it must have occurred very quickly. Gotterbarn added Barnes to the listserv not "early on" (as in 1994 or 1995) but on April 11, 1997 (after trying, and failing, to do it on April 7)—and that is the last we hear of him (except for his inclusion in the list of Version 2.1's "authors")—or rather, the last we hear of him until Gotterbarn and Rogerson deliberate about including him among the "authors" of Version 3 even though he contributed nothing to it and gave no indication of wanting his name listed. His name remained on all subsequent versions. Gotterbarn\Version 2-2a-2.1\Task force vote\SR1. See Chapter 8 for more details.

[45] Weckert, a Master of Divinity (1977), a Ph.D. in Philosophy (1984), and a Diploma in Computer Science (1994), taught computer science at Charles Sturt University in Wagga Wagga, New South Wales. He believes he first heard about the code from Gotterbarn at a conference in 1994 or a bit later ("work had already been going on for some time") but his name does not appear on any list (or other document) until April 7, 1997. His only disappointment in the process is not having participated more. Interview with Weckert, August 9, 2001.

[46] See Chapter 9 for the details.

[47] Gotterbarn\by year\1997\feb-ap 97\Feb 97 delivered item\df1. "Gotterbarn\History of SE Code\History expanded" gives the date of the Steering Committee's contacting him about the "disclaimer" (cover memo) as March 14, 1997. That seems to be a mistake. April 8 seems more likely since there is the email from Frailey describing the Committee's request that arrived on that date. Gotterbarn had every reason to work out the "disclaimer" as soon as possible, not let the issue hang fire for a month while he worked to arrange publication of the draft. And, of course, if we assume the earlier date, the Steering Committee's act would not be a quick response to Gotterbarn's "Status Report" (and the code that accompanied it).

[48] Gotterbarn\by year\1997\ feb-ap 97\feb delivered items\DF2.

[49] Comments on Chapter 7, September 22, 2003.

[50] Email, Miller to Weil (February 16, 2005).

[51] Email, Rogerson to Davis (March 17, 2005). Rogerson describes it as an "IEEE Certification of Appreciation for contributions to the development of the code of ethics for software engineering 1998".

[52] Version 2-2a-2.1\ver 2a\Hales2 (a newsy email, April 11, 1997 to Jim Hales at ETSU). This email also reports that, as a result of the trip, Gotterbarn had become "co-director of an international computer ethics conference to be held in Rotterdam [in March 1998]."

[53] Gotterbarn\by year\1997\ feb-ap 97\feb delivered items\DF2.

[54] Comments on Chapter 7, September 22, 2003.

[55] Richard G. Epstein, *The Case of the Killer Robot* (New York: John Wiley and Sons, 1996).

[56] Feldman declined to be interviewed for this book because (he said) he had had nothing to do with writing the code.

[57] Gotterbarn's files include a letter that his computer dates April 11, 1997 (addressed to "Dear Lori Clarke and William Tracz") explaining the joint IEEE-CS/ACM project and asking *SIGSOFT* (the publication of the ACM's Special Interest Group on Software Engineering) to "publish the draft of the code for review and comment by your membership". Gotterbarn Archive\Version 2-2a-2.1\Ver2a\SIGSOFT.

[58] Apparently, Gotterbarn does not distinguish between "2a", "2.a", and "2.0a". We shall do the same.

[59] It is hard to see how Gotterbarn could have derived this idea from the minutes of April 8, 1997. Perhaps there is a missing email.

[60] April 8, 1997. After thanking Gotterbarn "for the update", the writer votes "FOR distribution of this code in draft form for comment, both on websites and if possible in print." He then expresses his sorrow that "I am not in a position to provide [detailed commentary] in timely fashion." We can easily guess why Gotterbarn would not cout this as a "response".

[61] Gotterbarn\by year\1997\organizat\FIXLST.

[62] Comments on Chapter 7, September 22, 2003.

[63] Unfortunately, this email is dated April 24, 1997 (and indicates the email it is responding to also dates from April 24). That makes no sense at all, since the essay was, by that time, at the printers along with the rest of the code.

[64] Gotterbarn\version2-2a-2.1\delayed IFIP. A SIG is a subdivision of a Working Group (here WG 9.2), itself a subdivision of a Technical Committee (here TC 9).The author of the file is Jacques Berleur (whose work on codes of ethics Gotterbarn had cited in his February background paper on codes); the computer date on the file is October 7, 1997, suggesting that Gotterbarn may not have received the document until then. This is consistent with the minutes' concluding paragraph: "Jacques Berleur asked members of the SIG to send him any further comments on the code. He offered to be in contact with Don Gotterbarn and forward to him our observations." Apparently, not realizing how fast Gotterbarn was working, Berleur took several months to collect comments before writing up the minutes. By then, Berleur was a veteran of the recently failed attempt of the IFIP to enact its own code of ethics. For very interesting details, see Jacques Berleur and Marie d'Udekem-Geves, "Codes of Ethics: Conduct for Computer Societies: The Experience of IFIP", in *Technology and Ethics: A European Quest for Responsible*

Engineering, Philippe Goujon and Bertrand Hériard Dubreuil, eds. (Leuven, Belgium: Peeters, 2000), pp.327-350.

[65] Like Gotterbarn, the IFIP session seems to have missed (what should be) the obvious whistle-blowing provision (under PUBLIC): "2.01 Disclose to appropriate persons or authorities any actual or potential danger that they reasonably believe to be associated with the software or related documents on which they work, or are aware of, may pose to the user, a third party, or the environment." The label "whistleblowing" seems to be important for identifying a whistleblowing provision. Perhaps every clause should have a keyword (rather than only the principles).

[66] On August 7, 1997, in an email to "Dear Task Force Member" (and sent through the listserv), Gotterbarn reported: "In Corfu, the International Federation of Information Processing Working Group 9 viewed the code [2.1] quite positively." Gotterbarn could not have received the official minutes until two months later (October 7). He was here relying on what Jacque Berleur and Joseph Kizz had told him (that "it received good response"). Gotterbarn Chapter7cmt (September 30, 2004). What explains the difference between the early favorable (oral) report and the later much less favorable (written) report? The most likely explanation is that the written report, a summary of discussions rather than a series of votes, gave a false impression of the discussion. But there are others, for example, that, based on other experience with the group in question, Berleur and Kizz would consider even the written report to be positive.

[67] Why Weckert thought "useful" had to mean only "economic value" rather than just the opposite of "useless" is hard to guess. What is not hard to guess is that his alternative, "not harmful", would make an entirely different point. Software that did not work at all might be useless but not harmful.

[68] Comments on Chapter 7, September 22, 2003.

[69] How did Neusse get a copy of Version 2.1? He was not on any of Gotterbarn's lists; Version 2.1 had not yet been published on the web. Gotterbarn is not sure but notes that "once we had approval to publish 2.1 we were free to let others look at it." Comments on Chapter 7, September 22, 2003.

[70] It also has some recent defenders. See, for example, Heinz C. Luegenbiehl, "Themes for an International Code of Engineering Ethics", *Proceedings of the 2003 ASEE/WFEO International Colloquium* (American Society for Engineering Education), esp. pp. 8-9, http://www.asee.org/about/events/conferences/international/papers/upload/Themes-for-Int-l-Code-of-Eng-Ethics.pdf (September 22, 2004).

[71] A few days later (June 23, 1997), one of Mechler's friends (at IBM) emailed that he could not find www.computer.org and wondered whether Mechler had the address right. Mechler checked and reported that he had just tried the address using Netscape and it worked. A few minutes later the friend responded: "I tried another system and it worked! Mysterious computers.

I'll print it and comment. Thanks." Mysterious indeed! Who said machines have driven mystery from the world?

[72] Gotterbarn\History of SE Code\History expanded.

[73] Gotterbarn\History of SE Code\History expanded; and email of August 17, 2003.