Chapter 9: Back in the USA, Version 3

"It is impossible to make anything foolproof because fools are so ingenious."
—Murphy's Law #11

9.1 More comments on Version 2.1

When Gotterbarn returned to Johnson City in early July, 1997, his sabbatical officially over, SEEPP's prospects were much better than they had been the year before.[1] When he left east Tennessee for his autumn semester at George Washington University, he had all but given up trying to make SEEPP work, was focusing instead on his own working group, and was having trouble even with them. Of the nearly forty volunteers he had recruited since early 1995, only a half dozen or so remained at all active. His co-chair was no help. SEEPP had missed several deadlines for producing a code and, in fact, still had nothing on paper but a lot of task-force procedures.

That seemed a long time ago. Now Gotterbarn was looking forward to official publication of Version 2.1 of the code (8. Appendix) and planning for Version 3 (9. Appendix). The reorganized SEEPP was working well. Still, the six months ahead would not be easy. The schedule sent the Joint Steering Committee in February called for doing much before December. Gotterbarn was supposed to have finished the ballot to survey his task force concerning the code by July 4, but some work remained. By August 1, he was to have developed two "ancillary documents", one on "methods and techniques to support the code" and another on "special ethics areas". He was also supposed to have defined the process by which the Steering Committee would ultimately approve the code. (But that would have to wait until the Steering Committee was ready to think that far ahead.) By September 1, he was to have completed work on both Version 3 and a ballot for surveying software engineers who read the ACM's *Communications* or IEEE's *Computer*. There was still plenty of time for that. By October 1, the ballots were to have been published and returned. That date now looked unlikely. Publication schedules made November 15 a more reasonable deadline for return of ballots. There would then follow the complex but necessarily speedy revising of the code in light of the votes (and comments the ballot solicited), ending November 27, with a revised code (Version 3.x or perhaps 4) being submitted to the Steering Committee. December was to be used to revise that document taking into account Steering Committee comments. The final code and related documents were to be delivered on December 22. (By June 1998, the code was to be approved and dissemination to have begun!)[2] That December 22 date for completing work on the "final code" was, admittedly, ambitious—but (given the record of the first six months of 1997) not too ambitious.

Comments on Version 2.1 continued to come in. The first of these (July 10) was from Walter Elden, a practicing engineer long prominent in IEEE ethics activities, who was writing other members of the IEEE Ethics-Guidelines Discussion Group (to which Gotterbarn belonged).[3] Elden suggested that the Group ("we") look at 2.1 because it "is structured with both main statements of Code Ethical Behavior [and] supporting guidelines." He asked his group to consider "this [as] a possible model for creating a set of Guidelines for the IEEE Code of

Ethics". Elden wondered whether "[this being] an activity in which an IEEE entity (Computer Society) is participating" 2.1 should be "coordinated with the already established and possibly overriding IEEE Code of Ethics". He then told his group where to find the Software Engineering Code on the web. [4] The address he gave was neither the IEEE nor the ACM website, but that of DeMontfort's Centre for Computing and Social Responsibility. Why hadn't Elden used the IEEE-CS site?

The only reply to Elden's inquiry, or at least the only one to reach Gotterbarn, was sent eight days later by someone with a navsea (Naval Sea Systems Command) address. It was brief:

> I reviewed the draft and have some observations:
> a. The obvious one first; there is no mention of the IEEE code of Ethics.
> b. The "Product" should be the last Principle, not the first, as it is the least important one.
> c. There is no mention of not discriminating, etc.[5]

Coming across this email in Gotterbarn's archive, I must admit to feeling sorry for both Elden and Gotterbarn. Elden had asked whether the two codes *should* be coordinated. The email's author responded that they *are* not coordinated. Having thus restated the premise of Elden's question), he goes on to suggest revising Version 2.1 (rather than, as Elden had asked, providing ideas for the IEEE Guidelines). The first revision proposed, moving Product (Principle 1) from first place to last, makes two dubious assumptions: 1) that the Principles are (or should be) stated in order of importance (instead of, say, in an order putting first what is most likely to be needed or—something quite different—what is logically prior); and 2) that Product *is* the least important principle (when, for example, it could be argued that it is the most important since the point of organizing software engineering as a profession, including writing the code, is to improve the products software engineers make).

The second revision this email proposed (c) is what made me feel sorry for Gotterbarn. While it is true (as the email says) that there is no use of the word "discriminate" (or any of its transforms) in any clause Version 2.1, 2.1 does require managers to act fairly (Principle 5), to make fair ownership agreements (5.05), and to offer fair remuneration (5.07). Principle 7 similarly asks software engineers to treat fairly all those with whom they work (including, in 7.07, giving a fair hearing to the opinions of others). There is even a provision (2.05) that asks software engineers to: "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." What more would the double negative, "not discriminate" (or "not discriminate unfairly") add? How is a code writer to protect against such a failure to look beyond formulas to meaning?

On August 4, Langford, a member of the Task Force, responded to Version 2.1. He had gotten his copy from the ACM website. He had almost three pages of criticism, about equally divided between the Preamble and the body of the code. The criticisms were, as he said, generally of the "nit-picking" variety. Some just corrected typos (such as changing a capital "S" after a semicolon to a small "s"). Others were arguable matters of grammar (such as moving "fully" in 1.02 from after "to understand" to before "understand", thus splitting the infinitive, without any suggestion of why that might be a good idea). A few of the criticisms, when set side by side, suggest that even so careful (and sophisticated) a reader as Langford had trouble using the code. For example, Langford suggests revising 5.07 ("Offer only fair and just remuneration")

by deleting “only”, asking “Do we need ‘only’?” But he also suggests amending 4.04 (“Not knowingly use illegally obtained or retained software, especially on equipment of a client or employer or in work performed for a client or employer”) by appending “—or at all”. His reason for the addition? “This reads as though it is otherwise OK to use illegally obtained software!” Apparently, he is supposing that users of the code may reason that whatever 4.04 does not expressly forbid is permitted. Why then did he not think that “we need ‘only’” in 5.07 to prevent the suggestion that it is OK to offer unfair and unjust remuneration to some people as long as you also offer fair and just remuneration to some others.[6] Should not principles of interpretation be consistent across a single document? Should not anyone who has, like Langford, read through the Preamble with enough care to catch many typos have noticed that it contains (unusually) detailed instructions on how to interpret the code—instructions that should rule out the inference that what is not expressly forbidden is permitted (e.g., “The list of Principles and Clauses is not exhaustive…”)?

On August 5 (Tuesday), Gotterbarn sent an urgent email to his executive committee (“PLEASE ASAP”).[7] The email consisted of a short covering memo followed by “the ballot-survey to develop version 3” (which Gotterbarn says he wants to send the Task Force “by Friday”—a month behind schedule). The ballot itself consisted of another covering memo (addressed “Dear Task Force Member”), eighteen questions (about half asking only for a “yes” or “no”), and the text of the code (now “2.2”, after correction of typos). Several of the questions concerned Langford’s suggestions. The email asked the committee (that is, Miller and Rogerson) both to comment on the ballot (“and cast your vote!!”) and to respond to five requests for suggestions about how to word ballot questions (signaled by “**[”). The first of these (under ballot question 9) is typical: “In 4.09.. the unless clause needs something following it. Say ‘unless a higher ethical concern is being compromised?’ Then say what to do in that case? Confront the employer? **[suggestions here????]” (The ballot was, when completed, to consist only of yes-or-no questions.)

The last of Gotterbarn’s five requests differs from the others. Because it responds to an earlier query from Miller, it is implicitly directed only to Rogerson. The request (under question 17) notes that clause 7.08 (“Not take steps to supplant another software engineer after steps have been taken for employment”) is “a direct lift from engineering code”, but then continues (apparently in Miller’s voice): “I am not sure what the above item is aiming at. Not to try to take a job away from someone? Could the wording be made more explicit/clearer[?]” “DG” then responds in backets: “’supplant’ in Webster—to supersede, to take the place of through scheming, etc.” To which “KM” responds (now in brackets): “’Steps?’ I don’t get 7.08” Question 17 then continues (without brackets but in Miller’s voice): “In 7.08, awkward.. Not take steps?? Also the word supplant is ambiguous to me.. not even sure what that means here. Isn’t 7.08 risky?? Should we interfere when we see others violating the code?” Then comes the official request for suggestions: “this is intended to say that when another person has a job prospect or is negotiating a job that you won’t steal it from them. How can we fix the clause???”

Miller’s response to Gotterbarn’s call for help arrived, eleven pages long, that evening. Gotterbarn was not surprised. He had come to expect from Miller responses both prompt and detailed.[8] Miller voted, inserting comments as he voted, and then added further comments within the code that followed (a code already apparently containing earlier exchanges between him and Gotterbarn). Miller’s comments (and votes) suggest that very little remained to be decided. Miller voted for replacing “assure” with “ensure” buts adds “either is OK by me”. (Clearly,

Miller had little to do with the shift from “assure” to “ensure”.) For several questions, he simply voted yes, correcting one typo along the way. Then he voted no on question 8, suggesting that 4.04 might simply say, “Not knowingly use illegally obtained or retained software” (rather than, as the question proposed, “Not knowingly use illegal obtained or retained software, especially on equipment of a client or employer or in work performed for a client or employer”).[9] In response to Gotterbarn’s call for help with 4.09 (in question 9), Miller suggested adding “unless a higher ethical concern is being compromised; in that case, the employer should be informed of the engineer’s ethical concern.” Having voted yes on a few more questions, corrected another typo, and suggested some minor improvements, Miller forgot about voting on the remaining questions and just made suggestions. Clauses 5.08 (question 14) should read, “Not unjustly…job when the subordinate is qualified for the job.” Clause 7.02 (question 16) should read, “If another software engineer’s work is not in the public’s domain, review that work only with appropriate permission, provided this is consistent with safety.” Clause 7.08 (question 17) should read, “Don’t try to undermine another software engineer’s job prospects for your own personal gain.”

8.2 The v2 ballot

By August 7 (just two days after his call for help), Gotterbarn had received enough help to complete the ballot. Just before noon, he emailed it to the listserv—using his old George Washington address. Why did Gotterbarn not use his ETSU address? Every time Gotterbarn moved, his email address changed. The listserv would not recognize him at the new address until that address was added to the listserv’s addresses (with the old one being removed to avoid duplicate mailings). Gotterbarn had the listserv through a friend in the Chemistry Department at the University of Tennessee-Knoxville (UTK). When Gotterbarn needed an address change, he notified his friend who, in turn, notified UTK’s Information Service. They would then take care of the change when other work allowed. Some changes were made quickly; some took many weeks. (Updating Gotterbarn’s listserv was not a high priority at UTK.) Luckily, Gotterbarn’s old GWU address was both still active and still an address the listserv recognized.[10]

Gotterbarn began his email by reporting some good news about the code’s reception and congratulating “all” the task force for having worked “quite hard” and announced that “now I need two more things from you” because it “is time to develop version 3 of the Code”. He then directed attention to the “survey [below] to solicit your opinions about the suggested changes to version 2.1; noted that “we are on a tight schedule so please vote yes or no (but adding that “If you don’t like a phrase then please suggest a better one”); and explained the schedule a bit more:

> Version 3 of the Code is scheduled to be printed in IEEE Computer in October and the Communications of the ACM in November. The function of the publication is to solicit opinions on the Code from the memberships of the two societies. I need a positive response from each of you that you want your name associated with the code and listed officially in these publications as part of the task force that developed the code. If I don’t receive a positive response from you that you want your name listed [then] it will not be included in the list.

The covering memo ended with a compliment: “From all the comments I have received, you can be proud of your work on this!!!”

The ballot now had eighteen questions (followed by "v2.last" for both the Preamble and the body of the code). One question was titled "General"; the other seventeen, "Questions about the Principles". The general question concerned replacing "[the remaining occurrences of] the word 'assure' with the stronger word 'ensure'" in a) the Preamble and b) the Principles." For those who think ballot questions should be stated in neutral terms, this first question may seem to start the ballot off badly. The question had a clear bias. Those who wanted a "stronger" code were invited to vote yes on the general question. Those who favored uniformity but did not like "ensure" were given no opportunity, except comment, to change any "ensure" to "assure"; they could only prevent further changes from "assure" to "ensure". For those who, in addition, recalled the commentary on Version 2 in the triple email of February 10, just six months before (Chapter 7.5), the description of "ensure" as "stronger" than "assure" might have come as a surprise. After all, the commentary's argument for "ensure" had then been that "'ensure' is less legal than 'assure' which carries a formal guarantee and legal connotation (according to the OED); whilst 'ensure' means to strive to make things happen and no formal guarantee is implied."[11] Would not asking software engineers "to strive" for something require *less* than asking them to "guarantee" it? Had the executive committee's understanding of the meaning of "assure" and "ensure" changed over the six months since February (perhaps because of a visit to the OED)—or (as I think) was the preference for "ensure" over "assure" wholly independent of the reasons given, an instinct in search of a rationale?

Though the general question seems to promise a partisan ballot, most of the other questions are entirely free of editorial comment. In those few that are not, the comments seem accurate (and, in that respect, neutral—or, at least, neutral enough). For example, question 6, having asked whether 2.08 ("Feel free to donate professional skills to good causes") is too "open-ended and non-committal", continues (correctly), "It doesn't make it seem as if there is any obligation in this regard." Question 6 then puts to a vote language that seems to impose an obligation: "Donate professional skills to good causes when opportunities arise". Since Principle 2's "In particular" did not then limit obligations under it in any way, this change would, if adopted, demand quite a lot.[12] (Principle 2 read: "Software engineers shall, in their professional role, act only in ways consistent with the public safety, health and welfare.")

Within a few hours, Gotterbarn had his first response (sent direct), from Tom Jewett, Editor of *Computers and Society*, the publication of ACM's Special Interest Group on Computers and Society (SIGCAS).[13] Gotterbarn, who had known Jewett for years, was pleased, especially since Jewett began the response, "Delighted to be associated with this and on the list!!" Jewett was both someone who mattered in ACM and someone whom Gotterbarn respected. Jewett's response showed he had given the ballot some care. He voted no on question 1, commenting "'assure' is correct in this context [and] 'ensure' is already used where appropriate in v.2.0a." He also voted against splitting the infinitive in question 2 ("first version is grammatically correct").[14] He didn't care one way or the other about the change proposed in question 3, endorsed the change proposed in 4 ("puts it in the correct context"), and so on—until he voted no on question 18 (append to 8.08 "and we must do all we can to help our colleagues meet these guidelines"), commenting "change is vague, and a bit reminiscent of military academies").[15]

A second response arrived a few minutes later—from Mechler (using the listserv). Yes, Mechler wanted his name associated with the code. He thought it would be a good idea to use "titles" to "further validate the code". His would be "CCP" (Certified Computer Professional).

He had been doing what he could to get others to respond to the Version 2 posted on various websites. He had scanned the code and thought that "overall" it looked helpful, but he had several "comments". There was the usual question about "free of error" (Principle 1). "[F]or many systems," he observed, "that is not a cost-effective goal." He wondered about 1.01's "must have specs". For some projects, "that would be a mistake". Would it be a "mistake" because the clause requires that the specifications be "well documented" rather than, as in Version 1, simply put "in writing"? (Unless Mechler had changed his mind about the substance of the clause, that is the only explanation because the only other difference between Version 2.1's language and Version 1's is the cosmetic switch from "assure" to "ensure".) Mechler had similar practical doubts about 1.05 "Ensure an appropriate methodology" because "most se's don't have control over this". (Version 2.1 had dropped Verion 1's modifier "development" before "methodology". Was that the reason for Mechler's objection?) Clause 1.07's "ensure realistic estimates on projects" raised the same question: "again, most se's don't have control over this". (Did software engineers have control over, as Version 1 put it, "*proper* cost estimates" but not, as Version 2.1 had it, "*realistic* cost estimates"?)[16] Had the writers of Version 2 missed subtleties in the drafting of Version 1? Had they made the code more demanding when they thought they were only clarifying what was intended? Mechler concluded his preliminary comments by admitting that this was "the first I read the code for a while". Reading it now, he saw an omission in 8.01: "we seem to have left out 'analysis/requirements' to keep up with. We seem to have all of the life cycle except for the beginning."

Mechler ended his memo indicating that he had filled in his ballot by underlining "yes" or "no" (as appropriate). There was no underlining. A half hour later, Mechler emailed again: "My return did not have the underline so I [am] resending leaving in my answers [that is, deleting "no" if he voted "yes" or deleting "yes" if he voted "no"]". Mechler voted yes on all but three questions. He voted against dropping "only" in 5.07, against rearranging 7.02 (putting an "if clause" first), and against adding "and we must do all we can to help our colleagues meet these guidelines" to 8.08 (while agreeing that some such provision might be added).[17]

Over the next few hours, five more responses came in—from Fairweather, Kanko, Fulghum, Norman, and Prinzivalli (who also announced the birth of a grandchild).[18] Langford's response did not arrive until August 11 (using the listserv).[19] Laurie Werth (UT-Austin) also responded that day (using the listserv), not voting but indicating, "Yes, I would like to be included on the Task Force Report"—with a "ps" asking whether Gotterbarn had "any ideas about a workshop". Joyce Currie Little sent her vote in on August 18 (using the listserv). The last ballot to come in (or at least, the last we have) arrived on August 28, from me (using the listserv). Mine differed from the others in two ways. First, I said nothing about whether I wanted my name associated with the document. (All the others had said they did.) Second, I voted no on most questions. (Everyone else, except Werth, had voted yes on almost all questions.) The only questions on which I voted yes were 4 (adding "related to any work project" to 1.13); 6b (adding "and contribute to public education with respect to the discipline" in 2.08); 8 (replacing the long 4.04 with "not knowingly use illegally obtained or retained software"); and 13 (clarifying 5.08 so that it would read, "Not unjustly prevent a subordinate from taking a better job when the subordinate is qualified for the job").[20]

One might conclude from so many negative votes that I was unhappy with Version 2. There is some truth in that. I did feel much like the anonymous wit who defined a camel as "a horse designed by a committee". The camel is a useful animal, at least as useful as a horse, but it

is not elegant (that is, beautiful to look at and a pleasure to use). What I mourned in the transition from Version 1 to later versions was the loss of elegance. But I also understood that a living code must sooner or later pass into other hands, undergo changes of which its originator would not approve, and become a camel. In codes, the price for permanent elegance is death. I therefore appreciated what Gotterbarn was trying to do—fit the code to those who must use it and (above all) put it in a form that would cross the desert to ratification. There is even some (contemporary) evidence that this was my mood. About two weeks before sending my ballot in, I wrote Mechler (August 14, 1997):

> Your e-mail of August 8 (to Prof. Comp. Standards Task Force) reminded me that once upon a time you had offered to send me (on disk) all the e-mail correspondence you had concerning writing the code (so that I could make sure I had a complete file before I started writing about it).
>
> If that offer is still good, may I now take you up on it[?] Our work seems to be done—and I am itching to begin writing.

I understood that "our work", both Mechler's and mine, was (more or less) "done". Others were now in charge.[21] (Mechler responded two months later that he didn't "think our job was done since it [the code] is still being reviewed; maybe phase 1" but nonetheless soon sent the file.)[22]

9.3 Giving credit

While the votes were still coming in, Gotterbarn found himself with another problem of timing. As he reported to Miller and Rogerson on August 17—in an email with the Subject "Rapid response needed—what else is new": "I took two days off at the end of last week and during that time I received a note from CACM [*Communications of the ACM*] editor saying that they are working on the November Issue of CACM and need our document NOW (implication being get it in now or forget it)." Gotterbarn had discovered the message on Sunday and needed "to express mail the product [including Version 3] to her tomorrow—Monday afternoon!!!" Gotterbarn had attached the comments received by then (nine ballots), the vote for each question as it then stood (that is, minus Little's vote and mine), and as he had modified it taking into account both "the clear votes [of the task force] and some of my well-founded opinions :-)."[23]

Having noted that "we didn't get many responses", Gotterbarn posed a "Stupid little political question":

> The following did not respond: Peter Barnes, British Computer Society; Steve Barber esq., Douglas Phillips [another lawyer, one equally entitled to an "esq."], Patrick Sullivan, Computer Ethics Institute, S. Weisband, Sec. SIGCAS [Special Interest Group for Computers and Society]. Is it politically advantageous to retain any of these names or should I just delete them from the list of contributors?

The question Gotterbarn posed was not only political but ethical—and it was neither stupid nor little. Gotterbarn's August 7 email had explicitly said: "If I don't receive a positive response from you that you want your name listed [then] it will *not* be included in the list." (Italics mine.)

Those who had not responded to that email *may* have chosen not to respond because they relied on Gotterbarn's assurance that silence would be understood to mean, "Delete my name."[24] If any of them did rely on Gotterbarn's words, he now had a contractual obligation to do as he said he would do, delete their names. His reason for not deleting their names is (he admits) "political", the authority he hopes to derive from their names. That hoped-for authority also raises the question why others who did not vote (by this date) are not also candidates for having their name deleted. There were four (at this time): El-Kadi, Jayaram, Kallman, and Little. Some of these would have carried at least as much weight among software engineers as, say, Barber or Phillips (the two lawyers).

Having followed the ACM's innovation of listing names after a code, Gotterbarn (like the ACM) had to develop criteria for choosing whom to list. He in effect asked Miller and Rogerson for help doing that. (What, beyond express consent in response to an offer, should decide who is listed?) Rogerson's response (August 18) is disappointing: "IT COSTS US NOTHING to keep them in and it might give us a bit of extra credit"—credit to which "we" are (it seems) not entitled. Rogerson might instead have noted the short time allowed for the ballot (a ballot that had not set a deadline for response), the possibility that many of the silent may go on vacation in August, the unfairness of allowing mere silence (that is, perhaps, a failure of the email to reach its source) to count as consent, or even the importance of early contributions to SEEPP's work as a reason to ignore a mere failure to respond.[25]

Of course, the sudden acceleration of the publication schedule meant that Rogerson had only a few hours to think through and answer a rather long memo (thirteen pages). And Rogerson's life was not without its own difficulties. His response begins by explaining that he would have answered sooner but his home "telephone dial [is] in crash"; he had to drive to campus to "finish this off". His response, and Gotterbarn's original inquiry, nonetheless reveal a good deal about the drafting process.[26] It remains fluid and deliberative, even if rushed. Though giving the balloting considerable weight, the executive committee clearly does not feel bound by it. For example, Gotterbarn wrote that he "preferred the suggested revision of 5.05 [Jewett's] rather than the revision actually voted on—what is you[r] opinion of the change???" To which Rogerson responded, "Yes me too." Gotterbarn posed the same question for 5.08—except that there were two alternatives to the wording actually voted on. Fulghum had suggested shortening the clause to "Not unjustly prevent a subordinate from taking a better job"), while Norman had offered "Not unjustly prevent a subordinate from taking a better position for which the subordinate is suitably qualified." Rogerson thinks: "Go with no. 2."

Some of Gotterbarn's questions require more than a choice among two or three alternatives. "I am," Gotterbarn had written, "at a loss for 8.08" (question 18, adding "and we must do all we can to help our colleagues meet these guidelines" to "Consider violations of this code inconsistent with being a professional software engineer"). This was the only question in which the negatives won (3 yes's to 5 no's). Yet, there did not seem to be much hostility to the intention of the provision. Most of those who voted no had offered alternative wording. Gotterbarn wondered whether Rogerson had any suggestions for rewording.[27] Rogerson did: "How about…and encourage colleagues to adhere to this code (?at all times?)." Gotterbarn had a similar question about 8.01: "Ed made a good comment on 8.01, unless the word 'development' is broadly construed, we do not cover the analysis process….Any suggestions for a simple fix." Rogerson responds, "How about replacing phrase with… creation, testing and implementation[—]or rewrite as … further their knowledge of advances in software

development." Gotterbarn even raised questions about provisions the ballot did not cover. "In re-re-reading the code," he reports, "I got stuck at 4.05. What does '(is not inconsistent with)' mean in that imperative?????" Admitting that he too is unsure what 4.05 now means, he suggests rewording it: "4.05 Keep as confidential information gained in their professional work that is not in the public domain, where such confidentiality is NOT INconsistent with matters of public concern."[28]

That was Sunday. Monday afternoon, August 18, Gotterbarn sent off Version 3 to CACM. Along with the code itself went a short introductory essay ("Software Engineering Code of Ethics, Version 3.0") and a ballot. Gotterbarn, Miller, and Rogerson are listed as co-authors of the essay (and so, of the entire article). There were four paragraphs and one (long footnote). The first paragraph briefly sketched the history of and purpose of the Joint Steering Committee; the second explained the three task forces; the third explained that Version 3 was the work of one of the task forces, SEEPP (and "reviewed by the Steering Committee for distribution and comment"); and the last paragraph (after reporting that the code developed out of study of the codes listed in the footnote) continued (in language we can now recognize as stepping over many hard to see traps):

> All these codes try to educate and inspire the members of the professional group that adopts them. Codes also inform the public about the responsibilities that are important in the profession. Codes instruct practitioners about the standard 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 making. 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.

The ballot insert listed every provision of the code (except the Preamble), the Principles by number and key word, the subsidiary by number and first few words (enough for identification). To the left of each provision were five boxes, one each under "Strongly Favor", "Favor", "Uncertain", "Oppose", and "Strongly Oppose".[29]

The next day (August 19, 1997) Mechler emailed the listserv: "Don, Will the Code, new version, still be on the web site?" Gotterbarn responded that, while the new code would be on web sites, that would not be until "early October" because "I don't want to put v3 on the web sites too much before publication, because of potential version control problems." He concluded by promising to send Version 3 to the task force "shortly".

8.4 Budget anyone?

Since work on Version 3 was now complete, Gotterbarn could easily meet the deadline for its submission to other publications (and web pages). Until enough ballots had come back sometime in November, he could relax, that is, teach his classes and work on other projects, under no more pressure than an ordinary faculty member. He could even begin to think about what the Steering Committee should do once SEEPP had completed work on the code (as it was, according to the February schedule, to do by November 27). That, anyway, is what his September 5 email to Dennis Frailey seemed to say.[30] After noting that "life [is] easier" (and that

he had recently been elected "vice chair of SIG CAS and… to chair the ACM Professional Ethics Committee; a committee with $0 budget"), he reported:

> We are moving along with the code. One of the common questions is how to enforce the code. I think a prior question is the establishment of the approval process for the code. A simple procedure for approval would be for the Steering Committee to approve the Code when it is ready and then have each of the participating societies approve it. This would make possible the later adoption of the same code by other professional computing societies like the BCS by having them also approve it.[31]

Gotterbarn was, in effect, suggesting that the Steering Committee follow the procedure the ACM had used (modified to allow for the involvement of two societies instead of one). He was openly distancing himself from the IEEE's Standards process.

On September 8, Frailey replied, inserting his answers at (more or less) appropriate places within Gotterbarn's original. At the end of the paragraph quoted above, Frailey inserted a brief (non-committal): "All ACM and IEEE codes are strictly voluntary, although the societies have been known to look unfavorably on people who flagrantly violate their codes." And then, in reply to Gotterbarn's inquiry about "accessing the travel funding you set aside for the executive committee to meet and work on the revisions of the code", Frailey passed a piece of news Gotterbarn would later say hit him like a "shock wave"[32]:

> The steering committee's intent has been for all work to conclude this year. If the code is not ready for recommendation by then, it will have to go in the pile of incomplete work. I would need to know how spending such funds will contribute to completing things by the end of the year. [33]

Frailey later repeated the point (in answer to Gotterbarn's "What has been going on with this process, I have heard little about where the Steering Committee is at and what their plans are"): "The plan is to make a set of recommendations to the societies and cease operations at the end of the year." The recommendations will include "doing a more complete survey [for the Body of Knowledge] using paid staff" and "something TBD regarding education and the code of ethics". The "societies" (IEEE-CS and ACM) would then have to decide what, if anything, to do next. What is "really going on" is that the Steering Committee realized "that it cannot do all that must be done with volunteers."

As usual, the Steering Committee's focus was the Body of Knowledge. It was the only task force that would require a "more complete survey" and "paid staff". SEEPP had just about done its work—without paid staff (and, since February, more or less on schedule). Gotterbarn had not asked for more time, only for more planning for the code beyond its approval—a request repeating what his schedule had asked seven months before). The Steering Committee had not yet even thought enough about that suggestion to have any plans for 1998. Those plans were "TBD" (that is, to be decided *later*). Whatever was not done by the end of the year would have to go into "in the pile of incomplete work". The code would have no one to be officially responsible for getting it through whatever process ACM and IEEE-CS would require for approval, no one to see to its dissemination, and no one otherwise charged with helping it into practice. The code would be an orphan in a rough world.

Frailey was, however, not entirely negative. While the Joint Committee "is not planning to spend any more money unless it must be spent" and he (for one) is "disappointed that we have accomplished so little overall", he does not "wish to make matters worse". He recognizes that, while SEEPP "has spent the most", it has "probably done the most". So, if Gotterbarn has a "specific reason" for wanting money before the end of the year, he should say "what it is" and "what we can expect from it"—with the implication that he would try to get Gotterbarn the money.

Gotterbarn read Frailey's observation about SEEPP spending "the most" as it was intended. In fact, the Body of Knowledge Task Force had spent many times what SEEPP had spent. But the Body of Knowledge Task Force had not spent IEEE-CS or ACM money; it paid for its initial survey (and for preparations for the full survey) with money from the Defense Department. Ethics did not have a "sugar daddy" like that—or at least SEEPP had never found one. So, having made that mental adjustment, Gotterbarn was pleased to learn that SEEPP had "done the most"—despite the false starts and lack of resources.

Two days later (September 10), Gotterbarn wrote Frailey again.[34] More than a page long, this email is a relatively formal document, suggesting considerable time for its preparation. There are three numbered sections (after the introductory, "Your response raises several concerns"). Together the three sections made a case for funding one face-to-face meeting of the executive committee. But the first section seems to have another purpose, to get Gotterbarn a meeting with the Steering Committee "to finalize the critical stages of this vital project." He suggests around November 5-8 (when the IEEE Frontiers in Engineering Education Conference would meet in Pittsburgh). The first section of this email also reminds Frailey that "we" (presumably, SEEPP's executive committee) had in February sent the Steering Committee a plan calling for completing the Code by December "and having it recommended by the Steering Committee" and that "you" had earlier said that "I would get to meet with the Steering Committee to discuss…future plans." The second section explains how, with "a concerted effort", his task force could complete the code and related materials "by mid-December". He then asks when the last meeting of the Steering Committee will be. The third section, the longest, sums up the case for support for a (face-to-face) meeting of the executive committee: "Given the time constraint of the end-of-year termination of the Steering Committee it is imperative that the SEEPP executive committee get together to adequately address the issues that will arise from the members' responses [the membership polling on the Code]." Gotterbarn concludes: "You will recall that Simon Rogerson is already under pressure from his university to finalize his visit to the US for this project." He does not mention the amount involved. The amount Gotterbarn had budgeted (in February) was $1800 (for *two* meetings of the executive committee).[35]

We have no record of Frailey's response, but there must have been one. Gotterbarn's archives contain a letter in a file dated December 7, 1997 (Sunday). It is addressed, "Dear Dennis and Fellipe [sic]" and lists expenses in two columns, one labeled "ACM" and one labeled "IEEE-CS". All expenses for Rogerson are in the IEEE-CS column. All expenses for Gotterbarn are in the ACM column. And those for Miller are in both columns (though with most in ACM's). The letter begins by thanking "you" for making possible a "very productive" meeting of SEEPP's executive committee and expressing the hope that "the final version [of the code will go] to you early next week". One week after December 7 would be December 14. Gotterbarn was then telling his task force that he had a deadline of December 15 (Monday) to complete the code and

get it in to the Steering Committee. So, the file date of the letter seems very likely the date on which it was sent.[36]

That letter's second paragraph provides evidence that Frailey had responded to Gotterbarn's earlier budgetary request (and also explains the accounting's two columns). "You had," Gotterbarn wrote, "expressed concern that we bill the same amount for the meeting to each society." In order to do that, he had "had to split Miller's expenses". The letter ends with a suggestion that Cabrera was taking a more active role than usual in this expenditure (perhaps because it would be his last act as Steering Committee chair): "Because of your deadline Fillipe [sic], Rogerson has already sent in his receipts." Total expenses for the meeting were: $1093.—But I am getting ahead of my story.

On September 23, Gotterbarn received what seemed better news than Frailey's. David Notkin, then chair of ACM's Special Interest Group for Software Engineering (SIGSOFT), reported that he had "forwarded this [the 'Action Request' from the Steering Committee] to the SIGSOFT EC for comments."[37] That brief message indicated that cc's had also gone to the other two members of SEEPP's executive committee, to the chair and vice chair of the Joint Steering Committee (Cabrera and Frailey), to Barbacci (with no explanation), *and* to Gene Hoffnagle. Hoffnagle, then chair of the IEEE-CS equivalent of Notkin's SIGSOFT, was Notkin's rough equivalent within IEEE.[38] Apparently, the Steering Committee did not want to rely entirely on SEEPP's ballot for evaluation of the Code of Ethics, nor did it wish again to rely on its own resources alone. It had therefore decided—at the conference-call meeting brought on by Gotterbarn's urgent request for money (September 10)—to ask appropriate special interest groups to comment on Version 3.[39]

Hoffnagle's email to his organization's executive committee (September 24, 1997)[40] explains the process:

> Please take the time to review the proposed SE Code of Ethics and respond directly to the indicated addresses (cabrera@microsoft.com and frailey@dseg.ti.com) by the deadline (10/15). Failure to get our comments in will be taken as approval.

Hoffnagle then refers to one "attached note" that gives the website for the Code [Version 2.3], and another that "says there's an update and provides the latest version [3.0]." That attached note (addressed "Gene and TCSE") contains the following instructions:

> If reviewing this draft document, the following should be borne in mind: (1) Principles 2-9 are largely derived from existing Code of Ethics from other engineering disciplines[;] (2) We should ensure that Principle 1 in particular and the other principles as necessary are restricted to ethical issues. Those that are primarily technical issues in nature should be eliminated.[41]

The instructions come from an IEEE-CS-appointed member of the Joint Steering Committee, Leonard Tripp, not from the Steering Committee itself. Nonetheless, they may help us to understand better what the Steering Committee was trying to do.

The Steering Committee seems to have learned at least three lessons since the October 1996 exchange between Frailey, Shaw, and Mechler (5.6). The first is that the Steering Committee knew a lot less about engineering ethics than they had supposed. The Committee had

therefore decided to look for expertise elsewhere (among the leaders in software engineering). Second, Gotterbarn's February 1997 table had demonstrated beyond doubt that Version 1—and, by inference, its successor, Version 3—were "largely derived from existing Codes of Ethics from other engineering disciplines". Tripp therefore informed the code's potential critics of that so that they would not repeat the errors of the previous winter. The discussion had moved on. Third, the code itself was no longer controversial. The focus of criticism had become those provisions some members of the Steering Committee regarded as "primarily technical" rather than "ethical".

This distinction, common in debates about ethical codes, is seldom explained. Sometimes the distinction seems to be a matter of detail. Detailed provisions are "technical" (and belong in a "code of practice") while general provisions are "ethical" (and can be put into a code of ethics). Sometimes the distinction seems to be between those provisions which are "timeless" or "universal" (that is, ordinary morality) and those that are relative to time and place (much as laws are). Sometimes the distinction seems to be between those provisions that are (or at least seem likely to be) controversial (the technical) and those that are not (the ethical). What all these ways of distinguishing the ethical from the technical share is the assumption that technical standards are never ethical standards (or ethical standards, technical)—that form or substance distinguish one from the other. That assumption does not seem to rely on empirical evidence. Many codes of ethics, including codes of engineering ethics, are detailed and contain provisions that are plainly not universal, timeless, or uncontroversial. Rather than rely on a survey of actual codes of ethics, the assumption seems to rely on an unexamined definition of "ethics". It is therefore worth pointing out that there are definitions of "ethics" that do not support that assumption—for example, this one (explained in Chapter 1.4): *those morally permissible standards of conduct that all rational persons in a group (at their rational best) want all the rest to follow even if their doing so would mean having to do the same.* According to this definition, a code can be as detailed as can be (so long as the members of the group all want it to be), as relative to time and place (as members of the group want it to be), and as controversial as may be (because, for example, those opposed to it are not yet at their rational best).[42] Ethics is, according to this definition, not so much a matter of the content of rules (apart from moral permissibility) as the relation between the rules and the group they are supposed to govern. Ethical standards are not, like law, imposed on a group whatever it may want or, like morality, binding on members of the group because they are moral agents bound by them whether they belong to that group or not.

9.5 Notkin responds

Perhaps there was one more lesson that the Steering Committee had learned from the November 1996 debate—or its consequences. They then had before them a relatively finished document. They might, it seemed, have completed work on the code within a few months if they had followed an orderly procedure making small, perfecting amendments. Instead, the Steering Committee had improvised. Frailey and Shaw had made a great many criticisms of Version 1, most (as it turned out) unjustified. But one result of that criticism had (it seemed) been to send SEEPP back to the drawing board. The code had become larger, more complex, and more detailed; it had, in many small ways, responded to the criticism Frailey and Shaw had made. (5.6) But, on the whole, it was even less like what Frailey and Shaw desired. There was an irony in Frailey's lament about how long the process was taking. He had written the first critique

himself and thereby invited Shaw to write hers; he was, it seems, the author of much of the delay he was now lamenting.[43]

There is also an irony in Notkin's response to the Steering Committee. Notkin did not wait for his executive committee to work out a position. He responded on his own on September 26 (that is, three days after inviting his executive committee to respond—and probably just three days after receiving the invitation to respond).[44] His response, three pages long, begins with a cover memo (less than a page) addressed to "Felipe and Dennis (and Don)". After making clear that what follows are "my own comments" (and that he hopes that those of the executive committee "will be forthcoming"), Notkin nonetheless claims that because some of the comments "are (to me at least) quite serious", he will "state clearly that until some of these problems are clarified and/or fixed, that ACM SIGSOFT does 'not' endorse this draft code." He then (disarmingly) divides his comments into "two categories": general comments "about ethics (such as financial conflict) where I may be especially naïve"; and some about specific provisions where "I think the code is too naïve" (provisions concerned with property in software or the software industry). Notkin is "less worried about the first category than the second" because provisions in the second category are more likely to reduce significantly "the people that might pay attention to and learn from the code." He concludes this cover note by warning that his comments may sound "far more negative than they should" but hopes that (as a result) "they will generate some useful improvements".

SEEPP's executive committee seems to have taken Notkin's comments very seriously. Gotterbarn inserted his own responses in Notkin's email and then sent the amended document to the rest of the executive committee (late on September 26).[45] Miller replied early September 29 with comments inserted at appropriate points in Notkin's document, adding his two pages to Notkin's original three. Gotterbarn then forwarded Miller's comments to Rogerson with the instruction "Add your thoughts". Apparently, Rogerson had not yet responded to Gotterbarn's first request and, indeed, did not respond even to this until a month later (October 28). The email that resulted from passing Notkin's comments around in this way is an exchange in which Notkin sounds much like the preceeding November's Frailey-Shaw and Gotterbarn-Miller-Rogerson often sound much like Mechler.

On some questions, there seems to be a profound disagreement. For example, Notkin objects to 1.06:

> [Talking] about "effective procedure for promotion of quality and reduction of risk" [seems] reasonable, for sure. But what about a situation where a company (quite reasonably) chooses a niche in which their immediate goal is likely to be low quality but first on the market? Or what about a company that intentionally takes a high-risk strategy (perhaps, for example depending on other products that aren't yet completed)? Perhaps situations like these are covered indirectly in the draft code, but I don't believe that this is sufficient.

Gotterbarn throws up his hands (figuratively): "I don't know what to do here. If he is talking about deceiving the public, then it is not acceptable in a code of ethics. Is there a reasonable way to fit informed consent in here?" Miller adds that, while risks are "unavoidable and acceptable", they are acceptable "ONLY when the affected parties are able to choose that risk". Disclaimers

on risky software should be required. The bargain must be explicit. To which Rogerson can only respond, "I agree with Miller."[46]

On some issues, the disagreement, while still "serious", seems more nuanced. For example, Notkin had asserted:

> There is not a single mention of the issue of modifying or changing existing software. Since this is a piece of the industry, the omission is obvious and serious. I don't believe that adding in a few words can handle this easily…because in many cases the pressures to make changes without a "full" understanding of the software are real and reasonable.

For Notkin, the ability to "change and evolve software is one of the primary characteristics that distinguishes software from other engineering disciplines." If the code says nothing helpful about how to change existing software, "we can use 'any old engineering code of ethics'." Apparently, Notkin, a computer scientist, does not know that engineers generally examine the systems on which they work, looking for ways to improve them, and regularly do make improvements. (In fact, one way to distinguish between engineers and technicians (or, rather, "mere technicians") is that engineers are supposed to improve what they work on while technicians are merely supposed to keep what they work on as designed.)[47] Notkin does nonetheless raise an important question: why not just use an engineering code (a good one, not "any old one"); why write yet another code? The answer to that question would seem to be in (what was then) Principle 1, those provisions concerned in particular with "product".

That engineers might "change and evolve" their products also seems not to have occurred to the three computer scientists constituting SEEPP's executive committee. Instead, they follow Notkin in thinking about the question as one peculiar to software engineering. Miller wondered whether it "is 'reasonable' to make changes in code without a full understanding of the software???" Miller did not think it "reasonable or ethical" to change software without an adequate understanding of the consequences. Gotterbarn agreed with Notkin that "modifyability is one of the distinguishing characteristics" but not with the conclusion Notkin drew from it: "when we teach maintenance we say that the normal development process is applied to an existing software artifact—understand the needed change, design the change, implement and test the change[,] and make sure that the change ha[s] no unexpected side effects." Rogerson thought there might be a middle way: "Perhaps the issue is [whether] to undertake appropriate impact analysis prior to software modifications taking into account the role/ functionality of the software—the greater scope wrt [with respect to] people, organizations, society[,] the more care required. My ethical hotspot concept!"

Now and then, SEEPP's executive committee simply recognizes the merit of Notkin's comment. For example, Notkin observed that "1.14 talks about making tradeoff's visible to all parties concerned, including the public…[and] 6.13 talks about sharing 'useful software-related knowledge, inventions, or discoveries with the profession'." The draft does not, however, "make it clear that companies and software engineers have interests—in particular, certain classes of intellectual property—that needn't be disclosed, or that must be disclosed with care." Notkin recognized that the code does "address this somewhat in Principle 4, especially 4.05 ['Keep as confidential information gained in their professional work that is not in the public domain, where such confidentiality is not inconsistent with matters of public concern'], but explicitly observing this inherent set of conflicts seems of value." Gotterbarn's response to Notkin here is, "Any

suggestions for rewording." To this, Miller gives the opinion that it "seems doable with a few words" and then expresses "trust in your pen, Don." Rogerson adds: "guess we can fix this in Dec!!" (when they are to meet in person to revise the Code).

One set of comments is interesting simply for what it tells about how SEEPP's executive committee worked. Notkin objected to 6.07 ("only accept remuneration appropriate to professional qualifications or experience") that many of the top hundred richest people in technology "got rich essentially as software engineers who started companies." Notkin does not want to have to argue about whether they "accepted appropriate remuneration". Gotterbarn's response is surprisingly sympathetic. After observing that "[pointing] to the existence of robber barons to show that an imperative doesn't belong in a code does not seem reasonable", he admits that he "always had trouble with this clause." When *he* consulted, his price was "in part based on the criticality of the project and which elements of my talents are employed." He did accounting packages "at a cheap rate" but nuclear reactor development at a higher rate. "Same guy, but different talents are required." He then asked what the others thought. Miller is brief (and not his usually perceptive self): "Maybe we should make this more explicit: 'Never take inflated wages when you know it is a bribe." (If this interpretation were right, 6.07 would belong under Principle 3 JUDGMENT, not under Principle 6 PROFESSION, and would be redundant in 3.) Rogerson deepens the problem Gotterbarn had identified:

> I am reminded of my time in Poland when my Polish colleagues could not afford latest proprietary software because of global pricing policies from MS and the like designed to maximize profit and minimize administrative effort regardless of social/economic impact in developing countries. This is a difficult clause but I don't think its just about bribes[—] it is about social justice. A code should set the moral high ground for us to aspire to. Somehow we have to write a clause that does this but takes the RB [robber baron] supporters with us. December's "toughie" task!!!

That task was of their own making. In Version 1, the corresponding provision (5.07) was much more limited: "Only accept a salary appropriate to professional qualifications." That language had survived more than half a century in several engineering codes. The executive committee had substituted "remuneration" for "salary"—without explanation but apparently with the intention of extending the provision to consultants.[48] They knew nothing of the clause's history, that is, that it was intended primarily to keep engineers from accepting salaries inappropriate to (that is, below) their qualification. There was no risk of employers paying too much. The inequalities of the market saw to that (since engineers are supposed to be candid in any statement of qualifications).[49] Having inflated a workable provision into an unworkable one, SEEPP's executive committee would soon (Version 4) delete it altogether.[50]

While Gotterbarn, Miller, and Rogerson in this exchange with Notkin often sound like Mechler responding to Frailey and Shaw, Gotterbarn, Miller, and Rogerson did not make the mistake Mechler had. They did not send these first impressions to Notkin or the Steering Committee. Instead, they (that is, Gotterbarn with the assistance of Miller and Rogerson) carefully prepared a more diplomatic response about as long as Notkin's original email. The response seems not to have been sent until December 3. It began "Dear David (Dennis, Felipe)" and ended "Sincerely, Don Gotterbarn". [51] After thanking Notkin "for your careful evaluation of the draft Software Engineering Code of Ethics v3.0", Gotterbarn apologized for taking so long to

respond, but he wanted "to clear my response to your concerns with the rest of the [executive] committee that has been working on the code." He also let Notkin know that there have been other responses as a result of publishing the code and "our responses to your comments incorporate responses to other comments on the Code."[52]

The preliminaries over, Gotterbarn begins the substantive discussion: "You are right that we did not specifically mention maintenance". The committee did not intend any disrespect; it merely viewed maintenance as "another type of development". The committee nonetheless understood what was bothering Notkin, "the unfortunate circumstance you alluded to where some developers treat maintenance as a mindless exercise". That is an attitude the code should not, by silence, encourage. Gotterbarn therefore proposed to add new language to the Preamble "explicitly" mentioning maintenance (quoting two new sentences) and to revise 8.01 in much the same way. 8.01 would then read: "Further their knowledge of developments in the design, development, maintenance, and testing of software and related documents, together with the management of the development process."

Having given Notkin everything he could have wished on one point (almost a page and a half of you-are-right-about-maintenance), Gotterbarn takes up Notkin's next point, sounding as if he is again going to give Notkin everything he wants: "Your second comment about 'shrink wrapped software' development for the market, rather than development for a specific client[,] was right on the mark." Gotterbarn then announced that "we" are going to "stipulate a 'client' for shrink wrapped software: the person or persons who must be satisfied before the product is shipped." (Whether this stipulation is for purposes of discussion here or a change in the code itself is not clear.) Having defined his terms, he noted a "concern that a shrink wrapped client [sic] may agree to software that is damaging in some way to some users". Risks are, of course, "unavoidable and acceptable" but "only when the affected parties are able to choose that risk". There should therefore be disclaimers on the software. "Contractual bargains must be explicit."

Then, without even a new paragraph, Gotterbarn moved on to Notkin's "third point about those who 'choose a niche in which their immediate goal is to produce a software product that is likely to be low quality but first on the market." Tradeoffs will have to be made, but (Gotterbarn added) "the professional is obliged to make these clear. Look at the reaction to the Intel Pentium [chip] fiasco [bugs Intel knew about but did not reveal until the public discovered them on its own]."[53] Clause 1.14 ("promote maximum safety") will stay as it is.

Gotterbarn then returned to his accommodating tone. He agreed that Notkin is right that 6.13 ("Share useful software-related knowledge…) should be revised to respect the "rights of the third parties and confidentiality".[54] There should (he added) also be a cross reference to 4.05 ("Keep as confidential…"). Gotterbarn took a similar tack with 1.15 (after admitting, "We struggled with this"). 1.15 should be changed to (uppercase indicating the additions): "Work to follow SUFFICIENT industry standards WHEN AVAILABLE that are most appropriate to the task at hand….)". The word "sufficient" suggests that "there are several acceptable methodologies" and "when available" that there may be "no best method" for certain projects.[55] Clause 3.07 ("Refuse to participate in any decision of a governmental body….in which they, their employer, or their client has a financial interest") should be deleted because, as other commentators had also pointed out: the clause is too "absolute".[56] Its concern, however, is reasonable and already covered in 3.06 ("Disclose to all concerned parties those conflicts of interest that cannot reasonably be avoided or escaped and aspire to resolve them"). Finally, there is Clause 6.07 ("Only accept remuneration appropriate…"). It would, Gotterbarn explains, have

been better stated as “never take inflated wages when you know it is a bribe.” Apparently, that clause is to be revised accordingly.

Gotterbarn then concluded by observing how “very few comments [were received] from publication of [Version 2.1] in SIGSOFT this summer”, but now “we are…getting a good representative sample [in response to publication of Version 3.0]”. Like any “good software engineering project, more reviews improve the quality of the product.” SEEPP’s executive committee will be meeting soon to “put the finishing touches on the Code”. The committee “would appreciate further comments” from Notkin. Neither Notkin himself nor the SIGSOFT executive committee commented again on Version 3.[57] But, as Gotterbarn had told Notkin, he had by then a good many comments from others—and, as he not not tell Notkin, they seldom seemed concerned with what concerned him.

9.6 On to Version 4

One of the first comments to arrive after Nodkin’s, though addressed to “Gene and Leonard”, was a forward from Laurie Werth. Apparently, the author of the comment, “Carl”, was a member of the TCSE, the IEEE equivalent of Notkin’s SIGSOFT. In other respects, however, the response was quite different. The tone was, as Gotterbarn noted when (on October 14) he passed it on to Miller and Rogerson, “better… than the one from Notkin (ACM).”[58] Carl began by “[agreeing] with Leonard that these principles look very familiar.” Carl had taught computer ethics in a “CS program (2 hours, required course)”. The code “is a very good integration of all ethical principles.” His chief “trouble in reading [the code]” is “the so-called three-level classification of clauses: Aspire, Expect, and Demand.” He cannot “tell (with ease) which is which.” For example, Clause 3.03 says “Reject bribery”. Is this an aspiration, an expectation, or a demand? “It would,” he suggests (in a tone suggesting to me that he doubts that it can be done), “be a good service to us (educators) to label each clause one of the three levels if our dear authors can sort them out clearly.” He then adds, “There are still some typos I guess” and concludes that he is “excited about seeing [the code] and would love to use it in my course.” That was the last Gotterbarn heard from the TCSE.

For the Steering Committee, that silence would, presumably, count as support for the code (just as the silence of Notkin’s executive committee should). So, Version 3 had, by December, passed one test. The organizational critics of 3.0 had been few, in fact far fewer than Gotterbarn had expected. And only one of them, Notkin, had made any serious criticism of the code itself—and even his were limited to a few clauses that could easily be revised. The question, then, was: what would the ballots show? By December 3, when Gotterbarn sent off his response to Notkin, the answer to that question was clear: the changes the code required would be more than a change in the version’s decimal numbering could accommodate. There would be a Version 4.0.

9. Appendix (Version 3):

Software Engineering Code of Ethics

IEEE-CS/ACM Joint Task Force on Software Engineering Ethics and Professional Practices

Don Gotterbarn, Keith Miller, Simon Rogerson

In May of 1993, the Board of Governors of the IEEE-CS established a Steering Committee for evaluating, planning, and coordinating 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 steering 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 Software Engineering ethics and professional practices 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 as a whole, 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 3) was developed by the task force and reviewed by the Steering Committee for distribution and comment. The purpose of this distribution is to solicit comments from practitioners and other interested parties.

Codes, if carefully written and properly promoted, can be powerful instruments in the drive for Professionalism and in establishing safeguards for society. They do not have to be and should not be sterile which is often the perception that people have of them. This draft code evolved after

extensive study of several computing and engineering codes[1]. All these codes try to educate and inspire the members of the professional group that adopts the code. Codes also inform 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. These concepts have been adopted in the development of this code. 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.

---

[1] Codes refered to include: The American Association of Engineering Societies, Model Guide for Professional Conduct; Accreditation Board for Engineering Technology's, Code of Ethics for Engineers and Guidelines for The Fundamental Cannon of Ethics; The Association of Computing Machinery's Code of Ethics and Guidelines for Professional Conduct, The British Computer Society, Code of Conduct; The British Computer Society, Code of Practice; The Institute for the Certification of Computing Professionals; The Engineer's Council for Professional Development; Faith of the Engineer; The Institute of Electronical and Electronics Engineers, Code of Ethics; The National Society of Professional Engineers, Code of Ethics for Engineers, and the Project Management Institute Code of Ethics for the Project Management Profession.

PREAMBLE

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 or to cause harm, and to influence and enable others to do good or cause harm. To ensure, 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 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 professional software engineers share with all other human beings by virtue of their humanity. The second level obliges software engineering professionals to more challenging obligations than those required at level one. Level two obligations are required because professionals owe special care to people 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 and 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 three levels of professional obligation are recognized, 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 omission 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 addressed 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 whether your acts would be judged 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. However, 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; it provides 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

Principle 1: PRODUCT. Software engineers shall, insofar as possible, ensure 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 fully understand 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 or propose.
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. Whenever appropriate, delete outdated or flawed data
1.13. Work to identify, define and address ethical, economic, cultural, legal, and environmental issues related to any work project.
1.14. Promote maximum quality and minimum cost to the employer, the client, the user and the public. 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 to the user, a third party, or the environment, that they reasonably believe to be associated with software or related documents for which they are responsible, or merely know about.
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 caused by 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. Donate professional skills to good causes when opportunities arise and contribute to public education with respect to the discipline.
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. Refuse to participate in any 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. Ensure 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.
4.05. Keep as confidential information gained in their professional work that is not in the public domain, where such confidentiality is not inconsistent 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 a higher ethical concern is being compromised; then in that case the employer or another appropriate authority should be informed of the engineer’s ethical concern.

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. Ensure that employees are informed of standards before being held to them.
5.02. Ensure that 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 tempered with a desire to further that 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, processes, research, writing, or other intellectual property to which an employee has contributed.
5.06. Attract employees only by full and accurate description of the conditions of employment.
5.07. Offer fair and just remuneration.
5.08. Not unjustly prevent a subordinate from taking a better position for which the subordinate is suitably qualified.
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. Ensure 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 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 public health, safety, and welfare.
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 undermine another software engineer's job prospects for one's own personal gain.

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 analysis, 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.
8.07. Refrain from requiring or influencing others to undertake any action which involves a breach of this code.
8.08. Consider violations of this code inconsistent with being a professional software engineer and encourage colleagues to adhere to this code.

This draft Code was developed by the IEEE-CS/ACM 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,

---

Notes

[1] Gotterbarn does not recall the exact date of his return to Johnson City. What he does recall is that his wife left about two weeks before he did (in late June) because of the death of her mother. Gotterbarn followed "early in July". Email September 18, 2003 (RE: ch. 7). What we do know is consistent with what he recalls. On July 21, Ben Fairweather forwarded two emails sent to Gotterbarn's DeMontfort address. The earlier of these is dated July 10, suggesting a departure before then. Gotterbarn\Version 1\CODECOMMENT.

[2] Gotterbarn\Steering Committee\Schedule\SchedV2 (February 13, 1997).

[3] Elden was about to retire from the Harris Corporation after a forty-six year career in engineering (electronics, telemetry, data acquisition, communications, and networking). For a summary of his career, see: http://onlineethics.org/bios/elden.html.

[4] Gotterbarn\Version 1\CODECOMMENT. Why might Elden think that the IEEE code could override the Software Engineering Code? That is not clear. The IEEE code applies only to IEEE members (not to electrical and electronic engineers as such); it is therefore the code of ethics of a technical society, not of a profession. The Software Engineering Code applies to software engineers (and only to them), whether or not members of the IEEE; that is, it *is* a professional code (that is, a code governing members of the profession as such). Should the two codes conflict (something unlikely given the generality of the IEEE code), a software engineer would have to choose between his profession and remaining a member in good standing of the IEEE. But there is no reason why any profession should give priority to the code of ethics of a technical society to which its members belong, especially one in which its members are a minority (as software engineers were—and are—within the IEEE).

[5] Gotterbarn\Version 1\CODECOMMENT.

[6] Their reasoning would be (in effect): I follow the code because I follow it sometimes. The fallacy is obvious: to say you follow the code sometimes (but not all the time) is to say you violate the code sometimes (but not all the time). One does not have to violate a code often to act unethically. Indeed, one only has to violate it once.

[7] Gotterbarn\Version 2-2a-2.1\Taskforcevotes\Ballot.

[8] Gotterbarn Chap8cmt (September 28, 2004).

[9] When drafting Version 1, I tried to limit the obligations of software engineers to what they do *as* software engineers (primarily on the job, but also in public acts where they would be so identified). A professional code is, I believe, not a code for a whole life, but only for the professional part of it. One general direction of change from Version 1 to Version 3 is to restate obligations so that they restrict what software engineers do as private citizens in off-duty hours (as well as what they do when acting as software engineers). The proposed revision of 4.04 fits this pattern. Why then did I vote for the proposed revision (the one Langford suggested shortening)? Simply because the shorter clause was shorter—without loss of content. I was not

given the option of returning 4.04 to its original form: “Not knowingly use pirated software on equipment of a client or employer or in work performed for a client or employer.” I would, however, have been happy to replace “pirated software” with “illegally obtained or retained software”—as more exact (if also more wordy) than “pirated software”. One’s reasons for choosing one wording of a clause over another can depend on the most pedestrian considerations—and may not be obvious from the mere preference.

[10] Email (Gotterbarn) February 17, 2004. Gotterbarn wrote me about these troubles only a few months after they had been resolved (November 17, 1997). His description is worth quoting (Gotterbarn\Steering Committee\MIKENOV): “The technical confusion—someone else graciously let me use their listserv. As E-mail addresses change—your identity changes (a Humean problem:-)) and you get denied access to a list. As my E-mail address changes, I too get denied access. Since I am the guy empowered to change E-mail addresses on the list and my identity has changed, I need to E-mail the system managers at the University of Tennessee to change things.”

[11] February 10, 1997 (Principle 1, Comment 3).

[12] It did pass. As a result, Version 3 read: “In particular, software engineers shall: …2.08. Donate professional skills to good causes when opportunities arise and contribute to public education with respect to the discipline.” Why did they not divide this into two provisions (rather than paste them together with that “and”)?

[13] Jewett, who had become editor of SIGCAS the year before, taught computer science at California State University-Long Beach (including a course called “Computer, Ethics, and Society”). His specialty was databases and web design development. He nonetheless considered himself to be an engineer (with a “small e”)—based in part on an engineering degree collected in the course of an eclectic education: “I have a bachelor’s degree from the University of Illinois in music education (1965). I have a master’s degree in electrical and computer engineering from UC-Santa Barbara (1980) and a master’s in systems management (USC, 1987). And I have Ph.D. work (but no degree) in "Computers, Organizations, Policy and Society" [an interdisciplinary program at the UC-Irvine] in the late 1980s.” His first computer job was in 1971—with punched-paper-tape storage and program debugging in octal machine language—in his fourth year in the United States Air Force. (He enlisted in the Air Force after a draft notice ended a two-year career in education, as director of band and orchestra in a public school in Rockford, Illinois.) For a time, he served as a software acceptance-test leader and technical-manual writer. Later, he developed and advocated user requirements for advanced communication systems. At retirement from the Air Force (1987), he was program manager for development and acquisition of one segment of a major satellite system. Interview of Jewett, October 31, 2003.

[14] Editors seem to be more conservative about grammar in general—and split infinitives in particular—than most people. Jewett and I seem to have cast the only two no votes on question 2.

[15] Gotterbarn\Version 2-2a-2.1\Task force votes on 2.1 changes\votetom

[16] Following these comments is an unexplained email address for "Pete Coad" (probably pasted in by accident).

[17] Also in Gotterbarn\Version 2-2a-2.1\Task force votes on 2.1 changes\voteed.

[18] Gotterbarn\Version 2-2a-2.1\Task force votes on 2.1 changes\voteben (direct); August 8, 1997 (Kanko, using listserv, also in Gotterbarn\Version 2-2a-2.1\Task force votes on 2.1 changes\votemark); Gotterbarn\Version 2-2a-2.1\Task force votes on 2.1 changes\voteflug (direct); Gotterbarn\Version 2-2a-2.1\Task force votes on 2.1 changes\voteman (direct); and August 8, 1997 (Prinzivalli, using listserv).

[19] Also in Gotterbarn\Version 2-2a-2.1\Task force votes on 2.1 changes\votdunc.

[20] Two responses seem *not* to have survived: Ilene Burnstein's and John Weckert's. We know they must have existed because Gotterbarn cited them when he reported the final vote on September 17, 1997 (Gotterbarn\Version 2-2a-2.1\Task force votes on 2.1 changes\SR1). Burnstein may have used her ballot to correct her name. From here on she is "Ilene", not "C.S."

[21] There may be another factor contributing to both my dissatisfaction with the (minor) changes in the code and my sense of having passed it on to others. I was quite sick from February through May.

[22] October 10, 1997. This email (again) promised to send a disc containing the electronic version of his email files. Mechler also remarked that he had seen "your comments [presumably, my ballot] and wondered why your name isn't on the document?" If I had not noticed the omission earlier, I now must have. I believe it was about this time that I asked Gotterbarn why he had omitted my name (and Weil's). He responded that he had thought we wanted to be invisible, that we had never said we wanted to be listed, and that the message that went with Version 2.1 said we would be listed only if we said we wanted to be. But, if we now wanted to be listed, he would be happy to list us as soon as possible, that is, in (what became) Version 4. There is no paper record of this exchange. My memory is that it occurred during a phone conversation. Gotterbarn has the same memory but (wisely) wonders "if we are right". Gotterbarn Chap8cmt (September 28, 2004).

[23] Gotterbarn\Version 2-2a-2.1\Task force votes on 2.1 changes\SR1. This is not the original email but Rogerson's response (August 18, 1997) containing the original with his answers inserted.

[24] This is especially plausible for Barber who had resigned from SEEPP October 21, 1996, without doing any work on the code itself. Gotterbarn\Steering Committee\Barber. See also Interview of Barber, November 14, 2002.

[25] For Version 2.1, Gotterbarn had used the opposite approach, letting people take their name off by express request (April 24, 1997). Versions 3's list is missing one name present on

Version 2.1's, Mary Prior's. She specifically asked to have her name removed because she did not think she did enough to deserve listing. (Gotterbarn, having the work of others listed to compare with hers, is adamant that she did deserve listing.) Gotterbarn\People\Primary People.

[26] To the procedural aspects of process noted here, Gotterbarn would add something about the executive committee's spirit: "This level of commitment of the executive committee was typical—went to unreasonable efforts to respond and keep the process going—this level of effort was common and was a major contribution to finishing the project." Gotterbarn Chap8cmt (September 28, 2004).

[27] Gotterbarn's listing of respondents' comments under 18 includes one (Kanko's) which makes an important general point about the structure of the code (one the piecemeal process of amendment threatened to overlook):

> Note that the proposed change [in 8.08] doesn't fit in grammatically with the last phrase of Principle 8: namely, "…software engineers shall continually endeavor to:". That is to say, the last phrase in the body of any of the principles is really the beginning of the sentence that is completed by each of the numbered items after the body of the principle. So read the last line in the body of any of the principles and follow that with any of the numbered (e.g. 8.08) items. The result should be a complete, grammatically-correct sentence. Something to check in the other principles?

Kanko's point is easier to see if we actually write out the sentence as he instructs: "In particular, software engineers shall continually endeavor to:… 8.08 Consider violations of this code inconsistent with being a professional software engineer and encourage colleagues to adhere to this code." "Consider" is just too weak to follow the already weak "endeavor". Clause 8.08 seems to be the only example in this version of the problem Kanko identified.

[28] Version 2.1 had: "Keep as confidential information gained in their professional work that is not in the public domain (and not inconsistent), where such confidentiality is consistent with matters of public concern." (The parenthesized "and not inconsistent" looks like it was pasted in by mistake.) Compare with Version 1: "4.05. Keep as confidential information gained in their professional work that is not properly in the public domain."

[29] *Communications of the ACM* 40 (November 1997): 110-118. *Computer* 30 (October 1997): 88-92 also published the code and ballot but omitted them from the table of contents. They were republished in *Computer* 30 (November 1997): 88-92, this issue including them in the table of content.

[30] Gotterbarn's September 5 email to Frailey seems to have been a response to a request from Felipe Cabrera (the Steering Committee's chair) for a "status report". Whether this email is the status report itself, Gotterbarn's attempt to get the "lay of the land" before submitting the official report, or just another attempt to get money for a face-to-face meeting of the executive committee, is unclear. The Subject of the email ("Meeting of Executive Committee") suggests

the third. There are in addition at least three reasons to think it is not the official report. First, it is addressed to Frailey, not Cabrera; second, it seems too informal; and third, Gotterbarn's next email to Frailey (Dennis2) says that the "status report [indicated that] we are polling the membership of the IEEE-CS and ACM about the Code and will have the results in mid-November" but the September 5 email says nothing about polling. While Gotterbarn has no memory to answer these questions, he has a suggestion: "you must keep in mind that sometimes people press the reply button to a message that may be about X (status report) and the body of the message may be about something else. The convenience of pressing reply sometimes misleads one about the message content." Gotterbarn Chap8cmt (September 28, 2004).

[31] Gotterbarn\Steering Committee\Dennis11.

[32] Gotterbarn\History of SE Code\History Expanded. Gotterbarn does not think receiving the news from Frailey was unusual. His communication with the Steering Committee was generally through Frailey. Indeed, he can recall only one email from Cabrera during Cabrera's tenure as chair. Gotterbarn Chap8cmt (September 28, 2004).

[33] Gotterbarn\Steering Committee\Dennis11.

[34] Gotterbarn\Steering Committee\Dennis2. Though the salutation is "Dennis", the email's only (electronic) version is an email sent to Gotterbarn. One reason to believe that the email was sent to Frailey at about this time is that subsequent events seem to presuppose some such communication as this—and there is no evidence of any other.

[35] After Gotterbarn's signature is the announcement of "New Email and Phone Number". The new ETSU email may have added to his difficulties with the listserv.

[36] Gotterbarn\Version4\AABIL. This is probably the text of what became an email. Gotterbarn explicitly says in the second paragraph that "Each person is sending in a proper expense accounting, but I thought it would be helpful to let you see the total picture so you could clear any difficulty Miller might get into from the accountants [because he wants some money from each society]." The receipts would have had to accompany a letter sent by "classic mail"; there was then no way to send receipts by email. Gotterbarn must therefore have written the letter (and posted it) and then pasted it into an email (as a "heads up")

[37] Notkin, who was a member of the Department of Computer Science and Engineering, University of Washington, holds a BS from Brown University (1977) and a Ph.D. from Carnegie-Mellon (1984), both in Computer Science. He had just become SIGSOFT chair. http://www.cs.washington.edu/homes/notkin.

[38] Gene Hoffnagle, who held a BS in Mathematics from Case Institute of Technology (1967) and an MS in Computer Science from Johns Hopkins University (1976), worked as a researcher at IBM's Centers for Advanced Studies (Yorktown, New York). (www.computer.org/csinfo/BIOS/gene.htm). (He declined to be interviewed.) Apparently, Hoffnagle was copied

because he was then chair of IEEE-CS's Technical Council on Software Engineering (TCSE), succeeding Elliot Chikofsky.

[40] We seem not to have this email. I am reconstructing it from an email that Gotterbarn sent himself on October 14, 1997 (Gotterbarn\Version 3\Survey Comments\TCSE-1-5). That email consists in large part of an email Laurie Werth sent him on October 10, a pasting together of parts of at least two other emails: a note to "Gene and Leonard" from "Carl" (to be described below) and the note from Leonard (Tripp) to Gene and TSCE (already quoted). Tripp will become increasingly central to our story from now on.

[41] Gotterbarn's response to Tripp's instructions may be the first hint that he is thinking of re-titling the code: "This [Tripp's second instruction] is contrary to the committee interest in professional practices. Should we read the Code of Ethics to include the word 'Conduct' or 'Practice', e.g., 'Software Engineering Code of Ethics and Professional Practice'??" Rather than attempt to distinguish ethical issues from technical issues, Gotterbarn is suggesting using the title of the code to make the distinction unimportant. That is exactly what the ACM had done in labeling its 1990 code: "Code of Ethics and Professional Conduct".

[42] For more on this way of defining "ethics" (and for a corresponding universal definition of "morality"), see my: *Ethics and the University* (Routledge: London, 1999), Chapter 1; or *Ethics, Code, and Profession* (Ashgate: Aldershot, England, 2002). For more on what I mean by "rational best", see my "Brandt on Autonomy", in *Rationality and Rule-Utilitarianism*, ed. Brad Hooker (Westview Press: Boulder, CO, 1993), pp. 51-65.

[43] Dissatisfaction with the speed of the process seems to come primarily, if not entirely, from the ACM-appointed members of the Steering Committee (Feldman, Frailey, Boehm, Shaw, Zweben). I therefore wonder whether the ACM members may bring a different sense of time to the process than the IEEE-CS members. The ACM produced its code in about eighteen months. By that yardstick, the Joint Steering Committee's work was way behind schedule (thirty four months and counting). For the IEEE-CS members, on the other hand, the IEEE Standards-writing would be the yard stick. By that standard (average time for writing a standard seven years), SEEPP was well ahead of schedule. That Cabrera seems a much less active chair than Barbacci, leaving much of the day-to-day work to Frailey, may have made Frailey (in effect) the chair, adding to ACM's weight in deliberations during 1997.

[44] Gotterbarn\SEEP 1996-97\NOTOBJ.

[45] Gotterbarn\SEEP 1996-97\NOTSI (that September 26 email is at the end of this October 28 email).

[46] For one such criticism, the problem seems deeper. As Gotterbarn puts it, "How do we handle this? He knows he should not read the single bullet and that it is handled in [other] sections of 3, but then insists on reading 3.07 as a stand alone item."

[47] Michael Davis, *Thinking Like an Engineer* (Oxford University Press: New York, 1998). Here we see one disadvantage of having an executive board without a single member trained as an engineer. The sense of "technician" intended is, of course, that appropriate for categorizing a mechanic, lab technician, or the like (rather than in the sense in which anyone working in technology is a "technician").

[48] February 10, 1997: "5. replace 'salary' with 'remuneration'."

[49] See, for example, ABET Guidelines 5.f: "Engineers shall not falsify nor permit misrepresentation of their, or their associates', academic or professional qualifications. They shall not misrepresent nor exaggerate their degree of responsibility in or for the subject matter of prior assignments. Brochures or other presentations incident to the solicitation of employment shall not misrepresent pertinent facts concerning employers, employees, associates, joint ventures, or their past accomplishments with the intent and purpose of enhancing their qualifications and work."

[50] There seems to be a similar dynamic for other provisions Notkin criticized. Version 1's equivalent of 1.06 ("Ensure good management for any project on which they work, including effective procedures for promotion of quality and reduction of risk") was: "Assure proper management on any project on which they work, including proper procedures for control of quality and risk." The term "proper" left more room for interpretation, perhaps avoiding Notkin's concerns with niches. Version 1's equivalent of 1.14 ("Promote maximum quality and minimum cost to the employer, the client, the user, and the public. Make any tradeoffs clear to all parties concerned.") had been 1.13: "Promote maximum productivity and minimum cost to employer, customer, user, and public." There is no mention of making tradeoffs clear to anyone. None of these provision would survive into Version 4 (or, at least, survive in anything like their original form).

[51] Gotterbarn\Version 3\Survey Comments\replytonotk. The file date is 12/3/1997. The letter itself is not dated and gives no indication of having been sent as an email (except for the heading "Subject: SE Code comments". Gotterbarn's archives contain several drafts from the middle of November—with requests for comment from "Team": 1) Gotterbarn\SEEP 1996-97\TONOTRPL, identical to that in another file with the same date (Gotterbarn\SEEP 1996-96\ NOTRPLDN); and 2) Gotterbarn\SEEP 1996-97\NOTRPL, from the day before.

[52] For details concerning these responses, see next chapter.

[53] See, for example, D. Price, "Pentium FDIV flaw-lessons learned". *IEEE Micro* 15 (April 1995): 86-88.

[54] The letter actually (twice) refers to this clause as "6.15", but the language makes it clear that 6.13 is what is meant (and there is no 6.15). Notkin's original has the clause number right. Odd slip in an otherwise carefully prepared document.

[55] We are now a long way from Version 1's simple idea: "1.14. Avoid fads, departing from standard practices only when justified."

[56] In fact, 3.07 survived all the way through the final Version 5.2—in a somewhat revised form (as 4.06): "Refuse to participate, as members or advisors, in a governmental or professional body concerned with software related issues, in which they, their employers, or their clients have undisclosed potential conflicts of interest."

[57] That is, on Version 3. Notkin made one unofficial comment a year later on Version 5.2. Gotterbarn recalls: "In October 98 I gave a presentation on the Code at the SIGSOFT annual conference…where I met Notkin. At that point his only question was why we combined into one document a code of practice and a code of conduct." Gotterbarn email (February 22, 2004).

[58] Gotterbarn\Version 3\Survey Comments\TCSE1-5. This is the October 14, 1997 email by which Gotterbarn forwarded Werth's October 10 email to Miller and Rogerson. I am ignoring the comments contained in the Corfu meeting of the IFIP SIG9.2.2 (May 7, 1997) because, though they arrived about this time (October 7), they concerned Version 2, not Version 3, and have already been discussed in the preceding chapter. While they probably went into the mix of comments in Gotterbarn's head, they did not go into the summary of comments Gotterbarn drew up in December in preparation for revising Version 3; there is no evidence that Gotterbarn even passed them on to his executive committee. They were by now obsolete (as well as much more negative than he had supposed when he first reported to SEEPP what he had heard about them from the meeting's chair). (8.7)