# Code Making

## How Software Engineering Became a Profession

Michael Davis

Good to begin well, better to end well.”—Chinese fortune cookie

Center for the Study of Ethics in the Professions
Illinois Institute of Technology
Chicago, IL 60616
davism@iit.edu

**Table of Contents**

# Preface

"New systems generate new problems."
—Murphy's Technological Laws #17

## 0.1 An Important Event

On October 19, 1998, the Executive Council of the Association for Computing Machinery (ACM) approved a document titled "The Software Engineering Code of Ethics and Professional Practice". A few months later, the Board of Governors of the Computer Society of the Institute for Electrical and Electronic Engineers (IEEE-CS) did the same. That double approval successfully completed an undertaking officially begun almost six years before as part of a larger effort to "establish software engineering as a profession".

That double approval was an important event in both professional ethics and the history of professions. Software engineers design, construct, test, and maintain software. There are today at least a million "software engineers" around the world.[1] They already form a global occupation to an unusual degree even in this age of globalism. Software engineers daily work together across national borders and even across oceans. What a team of software engineers did in California during their workday may be passed to a team in India who, eight hours later, may pass it on to a team in Ireland who, at their day's end, return it to California. Because software can traverse vast distances almost instantaneously and at almost no cost, international cooperation is dependent only on technical infrastructure and justified trust in the technical skill and professionalism of those who must cooperate. With the increasing demand for software, especially for the complex software on which lives and property often depend, both the number of software engineers and their importance seem likely to increase substantially for decades to come. Theirs is a profession about which everyone should know. But theirs is especially a profession about which those who make technology or science policy should know. Much of what is, or will be, technologically possible depends on software, its cost, and the standards to which it can be built—that is, upon what software engineers do and how they do it.

Software engineering is also a profession that has provided us an opportunity to learn much about how an occupation becomes a profession. A great part of drafting the Software Engineering Code was done by email; and much of that email has survived. For many provisions of the Code, we need not guess how they came to be or what the drafters' intent was. We have a record of the stages the provisions passed through, with the drafters' explanation of each change. Though the record is incomplete (as records generally are), it is (as far as I can tell) fuller than any other we have for the writing of a profession's code of ethics. We have the materials for a "case study" of unusual depth, one of special interest for at least three reasons.

First, the Code is plainly not a mere variation of some other—certainly not of the IEEE's or ACM's. It is a large code (almost 3000 words). It has an unusually long preamble, many novel provisions, and some innovations in structure.

Second, the Code does not seem likely to be a mere academic exercise, no sooner adopted than forgotten. Computing societies outside North America (including some in China, India, and Europe) have adopted it. The code has already been translated into a half dozen languages.[2] A number of international corporations (e.g., Raytheon) have adopted it as a standard

of conduct for those working on software.[3] It has been included in a fair number of recent texts in computer ethics.[4]

Third, a "professional society", such as the American Medical Association or the American Bar Association, generally exists long before the corresponding professional code. The code is the work of one standing organization—if only, as with the Code of Ethics of the World Federation of Engineering Organizations, an organization consisting of other organizations. Software engineering has no such organization. The ACM and IEEE-CS generally operate independently. In many respects, they are competitors (even though the overlap in membership is substantial and the two organizations cooperate on many projects). Neither is a professional society (strictly speaking), that is, an organization the membership of which consists entirely (or even largely) of members of one profession. The membership of IEEE-CS includes many computer engineers (those who focus on computer *hardware* and "machine language"), computer scientists (for example, those interested in the mathematics of computing), and information systems specialists (graduates of business school interested only in the application of software). The membership of the ACM is equally diverse. Software engineers are a minority, though a large one, in both organizations. The cooperation of the ACM and IEEE-CS in writing the Software Engineering Code is therefore an interesting anomaly. But it is also an important anomaly. The ACM and IEEE-CS are not only large organizations in absolute terms; they are also the world's two largest organizations of those who make their living studying, engineering, managing, and teaching about software (though some of their members are merely interested in software). Each has close to a hundred thousand members, many outside North America.[5] What these two organizations do can have important consequences for computing-related activities everywhere.

Because the Code was written as part of a larger undertaking expressly designed to make "software engineering" a profession (whether a part of engineering or an independent profession was unclear), those interested in "professionalization" should find much in the history of the Software Engineering Code interesting. The two societies formed a "Joint Steering Committee for the Establishment of Software Engineering as a Profession" (the Steering Committee). The Steering Committee divided the work of making software engineering a profession among three task forces: one to define the body of knowledge; a second, to define the curriculum to teach that body of knowledge; and the third, to define a code of ethics that should guide application of the body of knowledge.[6] The task force concerned with the Code soon took the acronym "SEEPP" (for reasons explained in 3.4). Though my focus here is SEEPP, I could not avoid telling some of the story of the Steering Committee and of the other two task forces, much as one must speak of parents and siblings when recounting the life of an individual. Studying SEEPP from start to finish should, then, give insight into how one profession formed, why it formed, and why it adopted a code of ethics.

That is one reason I have written this book, to tell an important story. The second reason is to learn from that story how better to go about drafting a code of professional ethics. Several times a year, a professional society contacts the Center for the Study of Ethics in the Professions (CSEP) at the Illinois Institute of Technology to ask whether it can provide assistance with writing or revising their code of ethics. Because I am CSEP's "codes expert", I answer many of these inquiries. Over time, CSEP's librarian and I have turned much of what I used to say into a webpage with a small bibliography.[7] We add to it whenever we find anything that seems likely to be useful. This book should be an important addition to that bibliography. At last, we will be able

to offer would-be writers of a code the chance to see the problems that stand in the way of organizing such an undertaking, the way provisions get worked out, the interplay between big ideas and local politics, and so on. And, of courses, this book will be available to many who might not think to check our website.

The third reason I have written this book is to provide later generations of software engineers insight into how to interpret the Code—and how to change it—that can only come from what they call "documentation". The more we know about how a code came to be, the more we can appreciate its strengths and weaknesses, the compromises on which it rests, and the problems it solved or put off (and those the drafters never imagined). Knowing such things should help software engineers understand the Code and therefore help them to interpret it, to revise it when revising seems wise, and to fend off unnecessary revisions. This book should be as useful to software engineers trying to act ethically as to those (scholars, engineers, or anyone else) trying to understand professions.

## 0.2 Plan of the Book

This book has three main parts. Part One (chapters 2-6) describes events that led from the first use of the term "software engineer" to Version 1 of the Code. Part Two (chapters 7-10) describes the process of repeated revision by which what was Version 1 on January 1, 1997 had become Version 4 by year's end. Part Three (chapters 11-12) completes the story of writing the Code and sketches the Code's subsequent dissemination. In addition to these three parts, this Preface, and the usual Bibliography and Index, this book includes a short introductory chapter (Chapter 1) and an Epilogue. Chapter 1, designed for the specialist in professional ethics rather than for ordinary readers, explains the assumptions on which this book rests, especially its understanding of "ethics" and "profession". Most of the Epilogue is also designed for the specialist. It offers reflections on some ethical problems raised by writing this case study, by the method used to reconstruct what happened, and by the decision to create an archive for relevant documents. However, section 13.2 is for the general reader—or, at least, for anyone interested in drawing from this history practical lessons for writing a code of professional ethics.

Seven of the chapters include an appendix. Six of these are versions of the code corresponding to the version under discussion in the chapter (from the early Version 0 through the final Version 5.2). But one Appendix (Chapter 7) is a provision-by-provision table comparing Version 1 with twelve other codes of ethics (six from engineering and six from computing). These appendices should be useful for following events. This book is in part a biography of the Code (one covering "the early years"). There are therefore many references to particular provisions. While the provisions are (I hope) always quoted when needed, their context cannot always be and sometimes the context matters in ways hard to foresee. I have included these appendices in part to make it easier to follow discussion of particular provisions (and to see their context).

I have also included these appendices because I foresaw another use for them. This book is, in part, about writing a complex technical document. Those interested in technical writing may find much in the writing (and rewriting) of the code familiar—but not adequately caught in other studies of the writing of technical documents. Here are preserved, not all stages of the document, but at least a good many of them, along with the running commentary of authors and

critics. All the documents cited, and many not cited, are archived at the CSEP's library and at http://hum.iit.edu:8080/aire/mainindex.html.

This book is a work of history. History is not a chronicle ("one damn thing after another") but a narrative, that is, a description of events related in a way that makes sense (generally, because the later events seem to grow out of the earlier). Every history must have a first event and a last—as well as a relatively small number of events in between (whether arranged in simple chronological order or in some more complicated way). Anyone wishing to write history must choose among events. Without that choice, the history would be too large to write; and, being too large to write, not be history at all. Though there is no algorithm for choosing among events, some choices will be better than others, that is, provide a more comprehensive or comprehensible narrative; some choices will be worse; and some, merely different, emphasizing one interesting subject at the expense of others. Presupposing such a choice, no history can be more than one version of the past. Other versions are possible. No version is wrong, so long as consistent with the evidence we have. One version can, of course, be better than another for some purpose—or even overall.

Analogies between divinity and authorship are not without justification. Writing has its privileges. But, whatever the privileges, infallibility is not one of them. I have certainly made errors here, perhaps most often when I persevered in a conclusion against the advice of those whose judgment I respect (as I have now and then). I have therefore tried to provide enough detail to allow a reader to draw a different conclusion. I have, however, consigned some of the detail to endnotes to maintain narrative flow.

This book should be read through quickly the first time, ignoring the endnotes—and, for those who only want the story, ignoring the first chapter and epilogue as well. If the book seems worth reading again, even in part, it should be read more slowly, consulting the endnotes more or less as one might stop to read wall plaques while touring a strange city ("Here the Bastille once stood"). A good book is a battlefield where intelligent armies clash in the half light their big guns make overhead. Books die when they can no longer recruit readers into one or another of those armies.

## 0.3 Acknowledgements

While few books are written alone, this one is even more of a group effort than most. I came to the writing of history with few credentials. I was not trained as a historian. Indeed, the only history courses I took in college (too long ago) were in the history of philosophy and political thought. I never took a course even in the philosophy of history, much less in historiography or method. What I know about writing history I have learned by reading, by talking to historians (including my wife before she became a lawyer), and by trying my hand at a few small studies, at best a good way to *begin* learning a difficult art.[8] I have, therefore, benefited from having a historian on the board that advised me through four years of research and writing. A friend of long standing, Lew Erenberg, another historian, helped me find a suitable title for the book—and consoled me when I worried that I might not know what I was doing, telling me in effect: "No general's plan ever survived contact with the enemy."

I had a board of advisors to consult throughout this work: David Coogan, Humanities (Technical Writing), IIT (2001-2003); Donald Gotterbarn, Computer and Information Sciences, East Tennessee State University; Gerald Engel, Computer Science and Engineering, University

of Connecticut; Ilene Burnstein, Computer Science, IIT; Peter Whalley, Sociology and Anthropology, Loyola University of Chicago; Thomas Misa, Humanities (History), IIT; Ullica Segerstrale, Social Sciences, IIT; Vivian Weil, CSEP, IIT (chair); and Elizabeth Quinlan, CSEP's Librarian. In addition to what I gained from them individually (acknowledged elsewhere), there was one benefit that no one of them could provide alone. I learned much just listening to their debates concerning whether I should do "this" or "that". Sometimes competing opinions teach more than any one opinion, however sound.

I have found two members of the advisory board, the sociologists (Segerstrale and Whalley), especially helpful in understanding the limits of interviewing even for the purposes to which I eventually limited my interviewing and helpful as well in shaping interviews so that I could benefit in the limited way I came to consider appropriate. Though, in general, philosophers know even less about interviewing sources than about writing history, that was not my problem. I had learned the basics of interviewing during a few months as a reporter for my college newspaper (a career cut short by my inability to write news reports in a form the editor could use). In the late 1980s, I learned more about interviewing while carrying out sixty interviews as part of a study of relations betweens managers and engineers.[9] I conducted most of those interviews with Tom Calero, a veteran of many such studies, then a member of IIT's Stuart School of Business, since retired. That was good training for interviewing, but not for the interviewing I undertook for this book. Calero and I were not interested in the past so much as the present. We did not have to worry about the accuracy of memory.

Writing about the living has some advantages. Four of the living I studied (Burnstein, Engel, Gotterbarn, and Weil) were members of the advisory board. Like the other advisors, they had a duty to read what I wrote. Their comments certainly helped me make this a much more nuanced study than it would otherwise have been. Several other participants in writing the Code (people who had no such duty) also read one or more versions of the manuscript in part; and two (Ed Mechler and Keith Miller) read the whole thing. Like members of the advisory board, they corrected errors of spelling, grammar, style, and fact and challenged some of my interpretations. They also urged me to say more about some subjects about which I did not say enough or did not think to say anything. I have not always agreed with what they advised, but I have always been thankful for the advice. Advice would be command if we could not decline to follow it.

This book would have been impossible without two grants from the National Science Foundation (NSF). One, a small start-up grant, allowed CSEP's research group to follow events as they unfolded and to collect a basic archive during the crucial years 1995-97. A second grant (SES-0117471), much larger than the first (2000-2005) paid for the expenses of the interviews, including some in England, half of a sabbatical year, and another semester of writing rather than teaching (Spring 2004). It also made possible cooperation between CSEP and the Software Engineering Ethics Research Institute at East Tennessee State University, which transferred to CSEP many significant documents that might otherwise have been lost, including many documents from before 1995 or after 1997. Several interviewees, especially Ed Mechler and George Sigut, also saved important documents now safe in the archive. The second NSF grant has allowed Liz Quinlan, CSEP's Librarian, to oversee the archive, helping to put it in an electronic form I could use. (Her successors, especially Kelly Laas, have carried on the work.) In this, they have had the help of several of IIT's graduate students in computer science or engineering: Vishal Mehta, Vikram Modi, and Sushma Shankar, but especially Mayur Naik, Anoop Kabra , and Kapil Dikshit. That the archive is now "on line" is due in substantial part to

Ophir Frieder, Director of IIT Information Retrieval Lab (and IITRI Chair Professor of Computer Science at IIT), who is allowing use of his lab's AIRE (Advanced Information Retrieval Engine) application as our search engine. Jefferson Heard, doctoral student and lab member, did the work of coding and programming the engine to "crawl" our data.

I should also thank IIT for granting a sabbatical leave for the academic year 2002-2003, and my interviewees who, despite lapses of memory, not only taught me much about parts of the writing of the Code of which they knew more than I did but also about software engineering (and software engineers). I too had the help of a graduate student, Tony Spencer (Sociology, Northwestern University), who assisted at many of the early interviews—and whose untimely resignation I considered a substantial loss.[10]

The length of a book's acknowledgements is—or, at least, should be—a function not only of memory for debts owed but of the patience of readers and the publisher's budget. I have, I fear, now reached the limits of both.

Chicago, 2009

NOTES

[1.] There seems to be no hard number for "software engineers" (though I have heard estimates as high as 3,000,000 world-wide). The 1,000,000 used here is merely my conservative guess based on the opinions of those who seemed to have the best chance of being right. We are unlikely to have a better estimate until we have some way to track software engineers, not only those who graduate with the appropriate degree but also (what are still far more numerous) those who "convert" from computer science, engineering, or some other discipline some time in their career.

[2] http://seeri.etsu.edu/Codes/default.shtm (August 23, 2004).

[3] For a partial list of organizations that have adopted the code, see: seeri.etsu.edu/se_code_adopter/organizations.asp (April 17, 2004).

[4] See, for example: Sara Baase, *A Gift of Fire: Social, Legal, and Ethics Issues for Computers and the Internet*, 2nd edition (Pearson Education, Inc.: Upper Saddle River, NJ, 2003); Terrell Ward Bynum and Simon Rogerson, *Computer Ethics and Professional Responsibility* (Blackwell: Oxford, 2004); Michael J. Quinn, *Ethics for the Information Age* (Pearson Addison Wesley: Boston, 2004): Herman T. Tavani, *Ethics and Technology: Ethical Issues in an Age of Information and Communication Technology* (John Wiley and Sons: Hoboken, NJ, 2002). While Bynum-Rogerson (pp. 170-179) and Tavani (pp. 322-329) print the code complete, both Baase (pp. 439-445) and Quinn (pp. 370-379) omit the initial statement of principles ("Short Version") and the final credits (neither of which is necessary for teaching the code).

[5] On April 17, 2004, the IEEE-CS claimed 100,000 members; the ACM, 75,000. Deducting for duplicates, the total combined membership is (probably) between 125,000 and 150,000, a large number—but not as large as the 175,000 usually cited.

[6] Neither the Steering Committee nor the body of knowledge task force seems to have thought of the code of ethics as part of the body of knowledge—though, of course, it is (or, at least, should be).

[7] See www.iit.edu/departments/csep/PublicWWW/codes/bibliography (April 21, 2004).

[8] The most important of this work in history are: "What can we learn by looking for the first code of professional ethics?" *Theoretical Medicine and Bioethics* 24 (2003): 433-454; "Three Myths about Codes of Engineering Ethics", *IEEE Technology and Society Magazine* 20 (Fall 2001): 8-14 & 22; "Writing a Code of Ethics by E-Mail: Adventures with Software Engineers", *Science Communication* 21 (June 2000): 392-405; "Are 'Software Engineers' Engineers?" *Philosophy and the History of Science* 4 (October 1995): 1-24; "An Historical Introduction to Engineering Ethics", *Science and Engineering Ethics* 1 (January 1995): 33-48; "Righting the History of Mathematics, or How Sausage Was Made," *Mathematical Intelligencer* 16 (Fall 1994): 21-26; and "The Ethics Boom: What and Why", *Centennial Review* 34 (Spring 1990): 163-186. I do not count my work in the history of philosophy, not even my favorite (but

widely ignored): *Actual Social Contract and Political Obligation: A Philosopher's History through Locke* (Edwin Mellen Press: New York, 2002).

[9] The results of this research, funded by the Hitachi Foundation, are to be found in my "Better Communications between Engineers and Managers: Some Ways to Prevent Ethically Hard Choices", *Science and Engineering Ethic*s 3 (April 1997): 171-213.

[10] The original plan was for me to pair for interviews with a colleague in Social Sciences, a young political sociologist with considerable experience in interviewing, but he proved unexpectedly hard to replace after he left IIT unexpectedly. The graduate student was the best substitute we could arrange.

# Chapter 1: This History, Professions, and their Ethics[1]

Old men forget: yet all shall be forgot,
But he'll remember with advantages
What feats he did that day.
—Shakespeare, *Henry V*

## 1.1 Assumptions

In the chapters following this one, I assume that the Software Engineering Code of Ethics and Professional Practice is both a code of ethics and a professional code. I also assume that what I describe is the process by which software engineering became a profession (more or less). Those assumptions are controversial. On some widely-accepted conceptions of "profession", software engineering still lacks crucial attributes of profession, such as licensure or workplace autonomy; and on some widely-accepted conceptions of ethics, the content, form, or origin of the Code rules out its being "ethics".[2] Yet the story told here is not (primarily) about licensure, autonomy, or the like but about a code of ethics. To tell that as the story of an occupation becoming a profession (a story of "professionalization"), I must employ what are (still) controversial conceptions of ethics and profession. To admit controversy is, however, not to admit that these conceptions are *ad hoc* or without impressive defense. The evidence that they are not *ad hoc*, like their defense, is to be found in part in what follows—insofar as the story I tell relies on these conceptions and seems the better for doing so—but primarily in what I have written elsewhere.[3] This chapter is not the place to defend these conceptions; it is only the place to explain them in enough detail to assure that readers may read on without feeling lost either because they do not understand the conceptions at all or because they do not understand them well enough to conclude, however tentatively, that "there must be something to them".

## 1.2 Ethics as Special Standards

"Ethics" has at least five senses in ordinary English. In one, it is a mere synonym for ordinary morality, those universal standards of conduct that apply to moral agents simply because they are moral agents. Etymology fully justifies this first sense. The root for "ethics" ("ethos") is the Greek word for custom or character just as the root of "morality" ("mores") is the Latin word for it. Etymologically, "ethics" and "morality" are twins (as are "ethic" and "morale"). In this first sense of "ethics", software engineers could not have a distinct code of ethics because no one can. Since ethics is universal, it is the same for all. This sense of ethics would rule out the very undertaking we are to study. It cannot be the sense of "ethics" that concerns us.

In four other senses, "ethics" contrasts with "morality". In one, ethics is said to consist of those standards of conduct that moral agents *should* follow (what is sometimes also called "critical morality"); morality, in contrast, is said to consist of those standards that moral agents actually follow ("positive morality"). "Morality" in this sense is very close to its root "mores"; it can be unethical (in our first sense of "ethics"). "Morality" (in this sense) has a plural; each society or group can have its own moral code, indeed, even each individual can have her own.

There can be as many moralities as there are moral agents. You can have "your morality" (in which, say, abortion is wrong) and I can have "my morality" (in which it is not). But even so, ethics remains a standard common to everyone (or, at least, *may* be such a standard, depending on how one understands "critical morality"). The application of critical reason may (in time) yield a definite answer (say, that abortion is wrong, that it is not, or that it sometimes is and sometimes is not).

"Ethics" sometimes contrasts with "morality" in another way. *Morality* then consists of those standards every moral agent should follow. Morality is a universal minimum, our standard of moral right and wrong. Ethics, in contrast, concerns moral good, whatever is beyond the moral minimum (for example, "how we should live our lives"). Ethics (in this sense) is whatever is left of morality (in our first—universal—sense, which includes both the right and the good) once we subtract morality (in this third—minimum right-only—sense). Since (as we shall see) professional ethics consists (in large part at least) of moral *requirements*, this cannot be the sense of "ethics" with which we are concerned here.

There is, however, a connection between ethics in this sense and ethics in the sense that does concern us. What goes into a code of professional ethics should include some of ethics in this sense. Codes of professional ethics convert conduct that is morally good but optional for an ordinary person into a moral requirement for members of the profession in question. (I shall say a bit more about this in section 1.4.)

The second (or "should") sense of ethics is closely related to a fourth, a field of philosophy ("philosophical ethics"). When philosophers offer a course in "ethics" ("applied ethics" as well as "moral theory"), its subject is various attempts to understand morality (all or part of morality in our first sense) as a rational undertaking. Philosophers do not teach morality (in our first, second, or third sense)—except perhaps by inadvertence. They also generally do not teach critical morality, though the attempt to understand morality as a rational undertaking should lead students to dismiss some parts of morality (in its second, descriptive, sense) as irrational or to feel more committed to morality (in its first or third sense) because they can now see the point of it.

"Ethics" can be used in yet another sense, to refer to those *special, morally-permissible standards of conduct governing members of a group simply because they are members of that group*. In this sense, Hopi ethics are for Hopi and for no one else; business ethics, for people in business and for no one else; and legal ethics, for lawyers and for no one else. Ethics—in this sense—is relative even though morality is not. But ethics (in this sense) is not therefore mere mores (or ethos). Ethics must—by definition—be morally permissible. There can be no thieves' ethics or Nazi ethics, except with scare quotes around "ethics".

This fifth sense of "ethics" is, I think, the one implicit in the claim that one profession's ethics differs from another (or, at least, the sense that yields the most interesting interpretation of that claim). So, for example, while a philosophy course in Computer Ethics might differ from a philosophy course in Engineering Ethics in many ways, most of those differences would be irrelevant here. What is relevant is that the special standards governing software engineers can differ from those governing other engineers and other "computing professionals". Without this special-standards sense of ethics, what now seems an ordinary undertaking—the writing of the Software Engineering Code of Ethics—would be mysterious: Why would software engineers devote years to writing their own code of ethics? Why did they take so much trouble with the

exact wording of so many provisions? Why did they claim their code differed from others? And why does their code seem so different from others? We must sometimes dismiss the reports of participants as self-deception, misunderstanding, cover-up, or exaggeration, but we are not entitled to do that without strong evidence, evidence that I have found neither in the documents I have studied, nor in what participants have told me, nor even in what I myself saw. We must therefore assume that the sense of "ethics" relevant here is this special-standards sense.

Ethics in this sense resembles law insofar as both apply only to members of a specific group (rather than to everyone). Both are *special* standards. Ethics also resembles law insofar as both are *standards* to guide conduct (in prospect) and to judge it (in retrospect). Neither law nor ethics is a description of how people act (in the way a "scientific law" is a description). Both law and ethics tell us only how people should act. Having a standard is consistent with occasional violations. Indeed, we are unlikely to speak of standards, whether of law, ethics, or even skill, unless there are (or, at least, are likely to be) occasional violations. If, however, the violations become too numerous ("the rule rather than the exception"), the standard, whether legal or ethical, ceases to be an actual standard. It does not therefore die. The standard may live on as a "model" or "ideal". Ideal legal standards are standards that should be incorporated into legal practice. Ideal ethical standards are standards that members of the relevant group can recognize as what should (all else equal) be the actual standards of the group. The standards are *merely* ideal when they do not in fact govern practice. An ideal code of ethics is a possible or model code, a source of inspiration; an actual code of ethics, a living practice (as well as an ideal), a powerful claim on conscience.

These similarities between law and ethics do not preclude a fundamental difference. Law applies to people whatever they want. It originates in a formal authority (answering the question "Who is to say?") and carries with it means of enforcement (of which police, courts, and criminal punishment are the most prominent). Ethics (in our fifth sense) applies to members of the relevant group only because of something they want (the benefits of the voluntary practice the standards in question create and maintain). The claim that ethical standards have on conscience depends on what the standards actually say. There is no "ethics authority" in the way there is legal authority. Ethics is a standard of conduct that everyone in the group wants everyone else to follow even if their following it would mean having to do the same. The members of the group in question, all of them, are the ethical equivalent of the law's formal authority. A code of ethics binds those it governs in the way an ordinary promise binds its maker, not in the way a formal contract does.

## 1.3 Sociological Definitions of Profession

Distinguishing that fifth sense of ethics from the other four suggests the question: What is *professional* ethics? How do the special standards of professions differ from other special standards (if they do)? The answer to such questions depends on what we mean by "profession". Unfortunately, "profession" resembles "ethics" in having several senses. "Profession" can, for example, be used as a mere synonym for "occupation"—an occupation being any typically full-time activity defined in part by an easily recognizable body of knowledge, skill, and judgment (a "discipline") by which one can (and people

typically do) earn a living. It is in this sense that we may, without irony, speak of someone being a "professional thief". "Profession" can, instead, be used for any occupation one may openly admit to or profess, that is, an honest occupation: "Plumbing is a profession; thieving is not." "Profession" can also be used for a special kind of honest occupation (for example, "knowledge workers").

There are at least two approaches to defining this special kind of honest occupation. One approach, what we may call "the sociological", has its origin in the social sciences. Its language tends to be statistical. The definition does not purport to state necessary or sufficient conditions for some occupation to be a profession but merely what is true of "most professions", "the most important professions", "the most developed professions", or the like. Every sociologist concerned with professions seems to have a list of professions the definition must capture. Law and medicine are always on the list; the clergy, often; and other professions, such as architecture, accounting, or teaching, sometimes.[4]

We may distinguish three traditions in the sociology of professions (what we may call): the economic, the political, and the anthropological. Though individual sociologists often mix them in varying degrees, distinguishing them as "ideal types" here should help us to think about them more clearly, even in their less ideal (and more mixed) forms. What is wrong with all three ideal types, their failure to understand how central ethics (in our fifth sense) is to profession, remains even when the types are mixed.

The economic tradition interprets professions as primarily a means of controlling market forces for the benefit of the professionals themselves, that is, as a form of monopoly, guild, or labor union. The economic tradition has two branches: Marxist and free market. Among recent sociologists in the Marxist tradition, the best is still Magali Sarfatti Larson (*The Rise of Professionalism*, 1977); among sociologists in the free-market tradition, Andrew Abbott (*The System of Professions*, 1988) is a good example. For sociologist in this tradition (whether Marxist or free market), it is the would-be members of a profession who, by acting together under favorable conditions, create their monopoly, more or less forcing (or tricking) society into going along. Successful professions have high income, workplace autonomy, control of who can join, and so on; less successful professions lack some or all of these powers (more or less). These signs of success, like the monopoly itself, may be embedded in law, but need not be. What matters for the economic tradition are market arrangements ("economic realities"), not (mere) law. The success in question may be independent of what participants in events sought. The economic tradition delights in discovering "the invisible hand" at work.

For the political tradition, however, the law matters more. Often associated with Max Weber, the political tradition interprets profession as primarily a legal undertaking, a matter of (reasonably effective) laws that set standards of (advanced) education, require a license to practice, and impose discipline upon practitioners through formal (governmental) structures. To be a profession is to be an occupation bureaucratized in a certain way. For the political tradition, it is the society (the government) that creates professions out of occupations, and the society (the public) that benefits (whoever else may benefit as well). A recent work in this tradition is Robert Zussman's *Mechanics of the Middle Class* (1985). This tradition seems to have considerable influence among members of professions when they are trying to convince colleagues of the importance of supporting licensure: "We cannot be a profession if we are not licensed—as doctors and lawyers are."

The anthropological tradition, often associated with Emile Durkheim, interprets professions as primarily cultural facts, the natural expression of a certain social function under certain conditions. Neither the professionals nor society can have much to say about whether a certain occupation will be a profession. Professions are a function of special knowledge used in a certain way, a community created by a common occupation. Among recent sociologists, the best of those working in the anthropological tradition seems be to be Eliot Freidson (in, for example, *Professionalism: The Third Logic*, 2001).

Distinguishing these three traditions helps make the point that the sociological approach has not yet yielded a single definition of profession and, more importantly, is not likely to. Sociology's way of developing a definition, that is, abstracting from a (short) list of clear cases whatever is common to most or all, is unlikely to yield a single definition—or, at least, is unlikely to until sociologists agree on a list of clear cases sufficiently long to produce a plausible definition. Today, only two professions appear on all sociological lists. That is much too few to derive a plausible definition—or even a "statistically" reliable one. Whatever the utility of a particular sociological definition for a particular line of social research, no such definition is likely to seem definitive to more than a minority of sociologists. Why sociologists continue to generate definitions in this way seems to be a question best left to the history (or sociology) of sociology. It need not concern us further.

## 1.4 Professions and Codes

The other approach to defining "profession" is philosophical. A philosophical definition attempts to state necessary and sufficient conditions for an occupation to count as a profession. While a philosophical definition may leave the status of a small number of would-be professions unsettled, it should at least be able to explain (in a satisfying way) why those would-be professions are neither clearly professions nor clearly not professions. Philosophical definitions are sensitive to counter-example in a way sociological definitions are not. Philosophers cannot use the standard defense of sociologists: "I said 'most', not 'all'."

There are at least two kinds of philosophical definition. One, the Cartesian, answers the question, "What do *I* think a profession is?" It attempts to piece together in a coherent way the contents of one person's mind. There may be as many Cartesian conceptions of profession as there are people who ask themselves what they mean by "profession". The Cartesian method has no procedure for mediating between the definitions that individuals generate. Some of the definitions are startling.[5]

The conception of profession I shall be assuming here is not Cartesian but, as I like to call it, Socratic. It answers the question, "What do *we*, practitioners and philosophers, ('really') think a profession is?" Such a conception must be worked out through a conversation. The members of various professions say what they mean by "profession". Philosophers, or other practitioners, test those definitions with counter-examples, with consideration of consequences adopting the definition would have, and in other ways typical of philosophical examination of definitions. Any problems discovered in this way are fixed by revising the definition. The definition is again tested. And so the process continues until everyone participating in the conversation is satisfied that no problems remain. It is this "critical conversation" (what the Greeks sometimes called "dialectic") that underwrites the claim that the resulting definition is "what we *really* think a

profession is". After many years of applying this method, I have reached the following definition:

> **A profession is a number of individuals in the same occupation voluntarily organized to earn a living by openly serving a moral ideal in a morally-permissible way beyond what law, market, morality, and public opinion would otherwise require.**

According to this definition, a profession is a group undertaking. There can be no profession of one. The group must share an occupation. An occupation is something more general than a job description. Indeed, the term is useful only where it gathers together a family of job descriptions under a single heading while still distinguishing that family from others. So, for example, a group consisting of accountants and engineers cannot form a profession, though accountants can form one profession and engineers another. There is little or no movement possible from one family to the other (without a new degree). The discipline required of one is too different from that required of the other. That is what makes them distinct occupations. Engineers, on the other hand, do form an occupation, even though some engineers work in large firms and some in small, some in government and some in business, some overseeing the construction of bridges and some designing microprocessors. Movement between one field of engineering and another, one sort of practice and another, is still sufficiently easy (and common) for engineering to count as a single occupation. Much the same is true of accounting (with respect to its fields).[6]

Since what counts as a single occupation will depend on how much movement is possible between jobs of various descriptions, there will always be a question whether some group consists of one occupation or two. For example, do physicians and surgeons form one occupation or two? The answer to that question may depend on the reason it is asked. For purposes of membership in the AMA, physicians and surgeons are (today) one occupation. But for some purposes, say, the study of marriage patterns across occupations, it might make more sense to treat them as two.

To undertake to make software engineering a profession is to answer yes (however tentatively) to the question, "Is software engineering a single occupation?" To answer that way is, however, not to rule out important differences between the jobs software engineers hold, only to assert that the similarities are (on balance) more important than the differences for the purposes in view. Such an assertion of relative importance may well be controversial—within the occupation as well as outside of it. There is no way to resolve that controversy here. For us, what is important is that the controversy does not prove that software engineering is not (yet) a profession or even distinguish software engineering from "the true professions".[7]

According to the definition of profession to be used here, the group in question (the would-be profession) must organize its occupation to work in a morally permissible way. Where there is no morally permissible way to carry on the occupation, there can be no profession. There can, for example, be no profession of thieves or torturers (since living by theft or torture—in general, at least—is morally wrong).

The would-be profession cannot, however, rest content with avoiding acts that are morally wrong. A profession must set standards beyond what law, market, (ordinary) morality, and public opinion would otherwise require. That is, a professtion must set *special* standards.

Otherwise the occupation would remain nothing more than an honest way to earn a living (as plumbing is, for example). These special standards will be ethical (in our fifth sense of "ethics"). They will be morally permissible standards that apply to all members of the group simply because they are members of that group.

That professional ethics (in our first sense) applies to members of a profession simply because of that membership is no surprise. It is true by definition. What is surprising, I think, is that the standards in question (the profession's ethics) will be *morally binding* on every member of the profession simply because of that membership—and therefore ethical as well in both our first and second sense. Each profession is designed to serve a certain moral ideal, that is, to contribute to a state of affairs everyone (every rational person at her rational best) recognizes as good (that is, as what she wants to be). So, physicians have organized to cure the sick, comfort the dying, and protect the healthy from disease; lawyers, to help people obtain justice within the law; accountants, to represent financial information in ways both useful and accurate; and so on.

These moral ideals must be pursued openly; that is, physicians must declare themselves to be physicians, lawyers must declare themselves to be lawyers, accountants must declare themselves to be accountants, and so on. The members of a (would-be) profession must declare themselves to be members of that profession in order to earn their living by that profession. They cannot be hired as such-and-such (say, a psychologist) unless they let people know that they are such-and-such. If their profession has a good reputation for what it does, their declaration of membership will aid them in earning a living. People will seek their help. If, however, their profession has a bad reputation, their declaration of membership will be a disadvantage ("I am a phrenologist"). People will shun their help. In general, if the members of an occupation are free to declare themselves or not, they will declare themselves only if the declaration benefits them overall (that is, serves at least one purpose of their own at what seems a reasonable cost).

Where members of a profession declare their membership voluntarily, their way of pursuing the profession's moral ideal will be a moral obligation. They will, that is, have entered a voluntary, morally permissible cooperative practice (by declaring their membership in the profession—"I am an engineer"). If hired (in part) because of that declaration, they will be in position to have the benefits of the practice, employment as a member of that profession, because the employer sought a so-and-so and they declared themselves to be one. They will also be in position to take advantage of the practice by doing less than the standards of the practice require, even though the expectation that they would do what the standards require (because they declared the appropriate profession) is part of what won them employment. If cheating consists in violating the rules of a voluntary, morally permissible cooperative practice, then every member of a profession is in a position to cheat. Since, all else equal, cheating is morally wrong, every member of a profession has a moral obligation, all else equal, to do as the special standards of the profession require. "Professionalism" is (strictly speaking) simply acting as the standards of the (relevant) profession require. To be a "professional" (or "a real pro") is to be a member (in good standing) of the profession—or (by analogy) to act as if one were (that is, to act in the way the relevant standards require).

Like a promise, a profession's ethics—the special standards of the profession—impose moral obligations. Professional standards may, and generally do, vary from profession to profession. They are, at least in part, a function of opinion within the profession. Since opinions

vary, it is possible to have several professions sharing a single occupation, one profession distinguished from another only by its distinctive professional standards (arising from differences of opinion concerning important matters of practice). This is not a mere possibility. Professional standards, including somewhat different moral ideals, seem to be all that make physicians (MD's) one profession of medical healer and osteopaths (OD's) another.

The special standards of a profession generally appear in a range of documents, including standards of admission, practice, and discipline. A code of ethics is, however, a central feature of a profession, a statement of the most general standards of practice. So, for example, in the United States, publication of a formal code of ethics is the signal that an occupation has organized itself as a profession. A profession is organized insofar as these special standards are realized in the practice of its members, in what they do and how they evaluate one another.

Since formal codes of ethics were almost unknown outside English-speaking countries until well after the Second World War, some may object that this definition of profession is too "Anglo-centric".[8] I have two (compatible) answers to this objection. The first is the professions have a history much as does the steam engine or parliamentary democracy—and, like the steam engine and parliamentary democracy, much of that history seems to have occurred in English-speaking countries. For a time, the industrial revolution and parliamentary democracy were "Anglo-centric". Why, then, not professions too? Second, my point about codes of ethics concerned *formal* codes. In many countries lacking such a code, technical standards may incorporate the same standards a code of ethics would in England, Australia, or the United States (though implicit in details rather than explicit in the more general terms characteristic of a code of ethics). The code of ethics may, in this sense, be both in writing and still "unwritten". Whether the technical standards of a given country in fact serve as a code of ethics will depend on the attitude that members of the (candidate) profession generally take toward those standards (assuming the standards to be morally permissible). If they regard them as external impositions, they count as law, not as an (unwritten) code of ethics. If, however, they regard them as standards they want everyone else in the profession to follow even if that would mean having to do the same, the standards do constitute a code of ethics (even if an unusually detailed one and even if enacted as law).

An occupation's status as a profession is (more or less) independent of license, state-imposed monopoly, or other special legal intervention. Even a country with licensed attorneys may have no profession of law (just as the United States has licensed plumbers but no profession of plumbing). Indeed, professions should maintain a certain independence of law. While professions often commit themselves to obey the law, the commitment must be contingent. Insofar as the laws of a particular country are unjust (or otherwise fall below the moral minimum), any provision of a professional code purporting to bind members of the profession to obey the law (whatever the law says) would be void in that respect (just as a promise to do what morality forbids is void).[9]

## 1.5 Learning about professional ethics from history

To deserve to go down in history is a good thing, but actually to go down in it may not be. In common speech, "You're history" is bad news, the information that the person addressed is about to die—involuntarily and long before her biological time. Generally, to write history is

to write about the dead; the living belong to journalism and the social sciences. That was, however, not always so. When Thucydides wrote *The Peloponnesian War*, he wrote about events in which he had taken part. Even Herodotus, the "father of history", wrote his great work, *Persian Wars*, about events within living memory. It is with these Greeks in mind that I have written this case study. Though the story I tell is not about the collision of great armies or economic systems, events deciding the future of nations, it is nonetheless about a crisis in an important human institution (software engineering), a crisis within living memory. It is the story of how software engineering became a profession (in our preferred sense)—a story, not a sociological study, yet a story having significance for the sociology of professions—and for professional ethics.

One way to read this book is as an investigation of how well the sociological approach to profession fits what we know of software engineering. For me, what is striking is how poorly sociology's three ideal types (or any mixture of them) fits the story I tell. My characters are not (primarily) concerned with the conquest of markets, raising their income, or protecting themselves from competition; nor is there reason to believe that an unseen hand guided them to act as if they were so concerned. They were already doing well in the market—and expecting to do better. Though governmental intervention in the market, especially licensing, does have an important part in the story, its part is extrinsic, often threatening the movement toward profession. And, like most histories, mine is not (primarily) about cultural forces achieving the inevitable, but about contingent events, the interplay of personalities, and the place of words, organizations, and actions in human achievement. Of course, just as one cold day does not make a winter, so the failure of the sociological approach to fit one profession does not constitute a refutation. It does, however, constitute evidence against any theory of profession relying on one or more of sociology's three traditions. Sociological theories of profession should fit the professions they claim to explain. My story is a challenge to the sociology of professions as so far developed.

That challenge would not be important if we had many studies like this one, and most of the others pointed in the opposite direction. In fact, we have no others. Few (if any) professions have an archive for the writing of a code of ethics to match that saved for software engineering. The absence of such archives may explain—in part at least—why, so far, historians have paid so little attention to professional codes. It is hard to write history without documents. Another part of the explanation may be that historians of professions, lacking any alternative, have followed sociology in emphasizing other features associated with professions, features easier to document than writing a code. Nonetheless, the closest to a book-length study of a code of professional ethics before mine seems to be a collection of studies, *The American Medical Revolution*, describing the entire (fifty-year) career of the first code of ethics of the American Medical Association. That collection draws much the same conclusion I do. Its leading editor, Robert Baker, also uses recent work in the philosophy of professions to sharpen the questions his historians ask the records they have.[10] Our books seem to stand at the beginning of a new line of research.[11]

When I started this book, I thought of it as sociology (or journalism) rather like Kidder's *The Soul of a New Machine* (a classic study of engineers designing, building, and testing an early computer).[12] What made this book history rather than sociology, insofar as it is one rather than the other, is the evidence on which I came to rely. Much of the activity Kidder observed

consisted of people working with certain machines, conversations between those people, and so on. Most of what I observed were emails. Kidder augmented his observations with interviews, mostly interviews carried out while the people worked. Although I too interviewed participants, I interviewed them several years after the work I studied had ended. I did not, until then, have a grant to pay the expenses of interviewing. I could not even make a good case for such a grant until what I wished to study, the writing of an important code, had indeed produced a code likely to be important.

I therefore had documents of a sort Kidder did not, hundreds of emails. I did not have to depend on notes I had taken at the time. I had the events themselves, the emails (or, at least, printouts of them), piled a foot high on my desk. That was an advantage. I also had interviews of a sort Kidder did not, interviews conducted several years after the events they were to help reconstruct. That proved a disadvantage. I was often in a position to check my interviewee's memory against emails (and other documents). I soon realized that the emails were a much more complete and accurate record than anyone's memory, indeed, than all of the memories put together. I could not rely on interviews in the way Kidder had.

The Greeks wrote history about events within living memory (at least in part) because they lacked the archives that would have allowed them to write much about earlier events. For them, where memory, and the few surviving documents, ran out, history also ran out, and legend, myth, and fable (the playground of poets) began. Perhaps their memory was better than ours, or the events they studied, being more central to their lives, made a deeper impression, or perhaps much the Greek historians tell us did not happen—or did not happen in anything like the way they recount. I do not know. What I do know is that I have been deeply shaken by how much my interviewees forgot or misremembered. (They could report that they had never been reimbursed for travel when their own files showed they had; they could recall an important event that occurred in 1995 as occurring two years before; and so on.) I soon learned to bring a timeline to interviews (to prompt their memory and mine). Even so, I came to think that interviews can tell us much about what people are, what they think, and what they are doing, much less about what they were, thought, and did, even a few years before.

Though I first noticed this failure of memory in others, I soon noticed it in myself as well. Not only did I sometimes forget events indifferent or even unpleasant to recall, but some I was happy to recover as documents proved me better than I supposed. Whatever the law of memory, it is not simply self-interest or pleasure (the "advantage" Shakespeare alleges at the head of this chapter).There is no mechanical way to recover memory; indeed, often no safe way even to distinguish between true memory and false—apart from looking for harder evidence. Memory, even of recent events, is as much phantom as ghost. I have occasionally used interviews to fill gaps in the record, especially when two or more interviewees agreed on what happened, but I have preferred to use interviews to help me understand the documents. What memory seems to preserve best is the feel of things.

To write about the living is always to risk hurting someone's feelings and, these days at least, to risk a suit for libel as well. To write, as I have, about people most of whom are still in good enough health to care what I write, literate enough to read it, and sufficiently lacking in diffidence to challenge me any time they think I have gotten something wrong, may seem even more risky. Yet, the risks here are, I am happy to say, much lower than in most studies.

*Apollo 13* was an unusual success among movies. It lacked violence, sex, and even a villain. Its commercial appeal seems to have rested on nothing more than watching three moon-orbiting engineers, each imperfect enough to be likeable, successfully solve a complex practical problem (with the help of other engineers back on Earth). This book resembles *Apollo 13* in lacking violence, sex, and villains. Those looking for scandal will be disappointed. This is a story of intelligent people successfully solving a complex practical problem. Part of the dramatic interest is watching good people trying to do good things. There are many small failures of knowledge, skill, and judgment, something to be expected in any human undertaking—and, something rare, a great failure just avoided. Much of the rest of the dramatic interest is in the conflict of ideas. Many of the disagreements between my characters are about what a code of ethics is, how it should be written, and what it should contain. We can see how theories of profession affect practice—and how practice can support some theories and raise doubts about others.

NOTES

---

[1] Portions of this chapter were presented at a workshop, "Toward a Common Goal: Ethics Across the Professions", Sierra Health Foundation, Sacramento, California, August 26, 2006; to the Research Group of Ethics, Faculty of Letters, Hokkaido University, Sapporo, Japan, February 14, 2007; Second ASPCP International Conference on Philosophical Practice, Purdue University Calumet, Hammond, Indiana, May 19, 2007; Philosophy Section, Faculty of Technology, Policy and Management, University of Technology-Delft, The Netherlands, September 24, 2007; Center for Ethics and Technology, University of Technology-Twente, The Netherlands, September 27, 2007; and the Center for the Study of Ethics in Society, Western Michigan University, Kalamazoo, Michigan, October 4, 2007. Versions have also appeared in print: "How is a Profession of Engineering in China Possible?" [in Chinese] *Engineering Studies* 2007, 132-141; "Is Engineering a Profession Everywhere?" *Philosophia*, forthcoming; and Christopher Meyer, ed., *Journal Ethics* (Oxford University Press: New York, 2009), pp.-—-; and Elliot Cohn, Michael Davis, and Frederick Elliston, *Ethics and the Legal Profession, 2nd* (Prometheus Press: Buffalo, 2009), pp——.

[2] See, for example, John Ladd, "Collective and Individual Responsibility in Engineering: Some Questions", *IEEE Technology and Society Magazine* 1 (June 1982): 3-10.

[3] See, especially, Michael Davis, *Profession, Code, and Ethics* (Ashgate, Aldershot, England, 2002).

[4] For a better sense of the enormous variety of sociological definitions, see John Kultgen, *Ethics and Professionalism* (University of Pennsylvania Press: Philadelphia, 1988), especially, pp. 60-62.

[5] My favorite in this category is: John T. Sanders, "Honor among Thieves: Some Reflections on Codes of Professional Ethics", *Professional Ethics* 2 (Fall/Winter 1993): 83-103, in which "profession" is defined so as to include the *mafia*.

[6] I have, please note, chosen two occupations where licensure is optional (for most purposes). When licensure is mandatory, as in law or medicine, we are likely to suppose the license, not the discipline, creates the boundary.

[7] For those who think I should take a firmer position on whether software engineers form a profession, I can think of no better response than to quote David Hume, *Dialogues Concerning Natural Religion*, Part XII: "there is a species of controversy which, from the very nature of language and of human ideas, is involved in perpetual ambiguity, and can never, by any precaution or any definitions, be able to reach reasonable certainty or precision. These are the controversies concerning the degree of any quality or circumstance. Men may argue to all eternity whether *Hannibal* be a great, or a very great man…"

[8] I say "almost" because there were certainly some codes of ethics outside the English-speaking world before the Second World War. For example, the Japanese Society of Civil Engineers adopted its first code of ethics in 1938 ("Beliefs and Principles of Practice for Civil Engineers"). Perhaps if we looked, we would find many more such examples.

[9] I should like to thank Seumas Miller for questioning me until I saw the need to make this point. The point is, I take it, consistent with that made in Eugene Schlossberger, "Technology and Civil Disobedience: Why Engineers Have a Special Duty to Obey the Law", *Science and Engineering Ethics* 1 (June 1995): 163-168. While I doubt that engineers have the special duty that Schlossberger argues for, my point now is simply that its existence is independent of the claim I am making here. Engineers might both have that special duty and still have a reason to violate it if the laws are unjust enough.

[10] Robert B. Baker et al, *The American Medical Ethics Revolution: How the AMA's Code of Ethics has Transformed Physicians Relationships to Patients, Professionals, and Society* (Johns Hopkins University Press: Baltimore, 1999). For more about development of medical codes generally, see: Robert Baker, Dorothy Porter, and Roy Porter, editors, *The Codification of Medical Morality*, Vol. I (Kluwer Academic Press: Dordrecht, 1993), which covers the period before the writing of the AMA code, emphasizing developments in England in the eighteenth century; and Robert Baker, editor, *The Codification of Medical Morality*, Vol. II (Kluwer Academic Press: Dordrecht, 1995, covering the nineteenth (and early twentieth) century, with an emphasis on the United States.

[11] It is surprising how little has changed since Ivan Waddington, "The Development of Medical Ethics—A Sociological Analysis", *Medical History* 19 (January 1975): 36-51, observed in his opening paragraph: "It is a curious fact that despite the rapidly growing volume of literature on professions, little work has been done by sociologists on the development of professional ethics. This omission becomes doubly curious when one considers the central importance attributed to professional ethics in much of the literature on professions".

[12] Tracy Kidder, *The Soul of a New Machine* (Atlantic-Little, Brown: Boston, 1981). Perhaps it is worth pointing out that Kidder *is* a journalist.

# Chapter 2: Before SEEPP, 1968-1994

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

## 2.1 NATO, DoD, and SEI

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

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

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

bad to each other and to those who increasingly asked them for large programs, programs too large for one or even a few programmers to write.[6]

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

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

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

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

for educating future software engineers. SEI even helped to develop guidelines for a course on the ethics of software engineering.[7]

## 2.2 New Jersey, Licensing, and Fletcher Buckley

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

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

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

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

software", and even advertise themselves as "software designers", "software developers", or "software architects".

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

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

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

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

1994 and as Chair of the Software Engineering Standards Committee from 1981 through 1983. In 1984, he became the Computer Society's first Vice President for Standards Activities. He had also served as a member of the IEEE-CS Board of Governors (1984-1986), initiated five IEEE software engineering standards seminars, and received several IEEE awards.[15] Buckley was a software engineer's software engineer.

## 2.3 Buckley Proposes

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

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

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

> We have received a couple of replies to our long memo that indicate that the reader believes that Laurie or I authored the motion. In fact the motion is from Fletcher Buckley, a person from outside the ACM Ed Board, the IEEE-CS Educational Activities Board, the SEI or the executive committee of TCSE. In short it is an independent effort, but one

> with enough steam behind it that the education community needs to think about its response. Let us have your thoughts and your references.[16]

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

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

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

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

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

Second, according to paragraph (c), the new profession, even though a new engineering profession, was to have its own code of ethics. For Buckley, apparently, having a code of ethics is central to being a profession. That was a proposition with which most engineers might agree. But Buckley seemed to assume that the new profession's code would not be the IEEE's or ACM's but something distinct from both. Since both the ACM and the IEEE had recently adopted new codes of ethics, codes with which (presumably) they were still happy, Buckley's assumption that yet another code of ethics was needed seemed to define software engineering as a new profession distinct from both computer science and other parts of engineering, inviting opposition from within IEEE as well as from ACM.

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

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

> LAST WEEK I SENT OUT AN ANNOUNCEMENT OF A FORMAL PROPOSAL FOR A STANDARD TO LICENSE SOFTWARE PROFESSIONALS. THIS PROPOSAL WILL BE PRESENTED TO THE IEEE BOARD OF GOV.S AT THE ICSE [International Conference on Software Engineering] CONFERENCE IN BALTIMORE. AN INFORMAL LUNCH ON FRIDAY 21 MAY AT 12.15 IS PLANNED TO DISCUSS LICENSING AND CERTIFICATION. THE LUNCH IS NOT SPONSORED BY ICSE, BUT BY SOME PEOPLE INTERESTED IN LICENSING & CERTIFIC ATION.

WE PLAN TO MEET AT THE NICKEL CITY GRILL (201 EAST PRATT STREET, BALTIMORE 21202, 410 752 0900)

I WILL REPORT TO THE LIST ON THIS AND OTHER DISCUSSIONS ABOUT LICENSING AND CERTIFICATION.

DON GOTTERBARN

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

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

## 2.4 A substitute motion approved

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

> That the Board of Governors of the IEEE Computer Society establish an ad hoc committee, appointed by the president, to serve as a steering group for action, planning, evaluation, and coordination related to establishing software engineering as a profession, subject to:
>
> > 1. The ad hoc committee (the steering group) will be composed of well-respected individuals (such as IEEE Fellows from the Computer Society's software engineering community, others of similar standing, and representatives of involved Computer Society units) with a balance of industry, research, and academic backgrounds.

> 2. The committee will coordinate on this issue with the various Computer Society boards and committees, other professional societies, and other communities.
>
> 3. The committee will establish and appoint task forces and working groups as necessary to accomplish the committee's work, with membership open to interested parties including non-members of the Computer Society and the IEEE.
>
> 4. The various proposals presently submitted are referred to the committee for appropriate action to advance software engineering as a profession. [The committee shall] Consider and document the issues associated with software engineering as a profession, including, but not limited to:
>
> > The factors involved in, and the value thereof, of establishing software engineering as an approved program including the associated accreditation issues.
> >
> > The factors involved in, and value thereof, of establishing a separate set of software engineering ethics.
> >
> > The factors involved in, and the value thereof, of establishing software engineering as a certified or registered field.
>
> 5. The committee is charged to report on its initial plan and program at the time of the November 1993 Board of Governors meeting.[21]

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

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

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

On May 18, Chikofsky had chaired a meeting of TCSE's executive committee at which Buckley explained his motion. Chikofsky had invited Buckley because he wanted to see how Buckley proposed to respond to what Chikofsky thought was a deep split between computer science and engineering on the issue of making software engineering a profession. While Chikofsky agreed that software engineering should be a profession, he did not favor trying to make it one if the attempt would split the software engineering community.[25]

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

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

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

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

## 2.5 The Steering Committee begins work

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

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

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

> Council endorses the establishment of a Commission on Software Engineering to address the question contained in Council backup, Item 3.7, Pages 3-4 of 7. If possible, this activity shall be done jointly with the IEEE Computer Society. The Executive Committee shall appoint (ACM's) members of the commission, provide it with a reasonable expense budget, and take other steps as may be necessary to assure that the commission may complete its work in timely fashion.

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

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

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

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

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

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

## 2.6 The Steering Committee's Four Recommendations

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

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

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

The steering committee's second recommendation was "identification of a required body of knowledge and recommended practices". Recognizing that the knowledge of software engineering was evolving, the committee recommended that the task of identifying the body of knowledge be carried out primarily by "industry experts", those likely to be most up to date. This recommendation was a departure from Buckley's original motion. His recommendation (c) was

for the steering committee itself to identify the body of knowledge in cooperation with the IEEE-CS Educational Activities Board. Apparently, the steering committee came to believe that defining the body of knowledge was too big and technical an undertaking for it or a few academics to undertake. Nonetheless, as events were to show, the steering committee drastically *under*estimated the difficulty of defining the body of knowledge.

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

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

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

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

The report's "Epilog" indicated that the committee would take up licensing, certification, and other "regulatory instruments" in "our next report". The committee's next report (March 4,

1994) did not do that. Instead, it merely described progress on the four recommendations discussed above. The progress suggests substantial change in how the committee saw its work:

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

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

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

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

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

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

Notes

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

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

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

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

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

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

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

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

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

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

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

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

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

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

[13] Gary Ford and Norman E. Gibbs, *A Mature Profession of Software Engineering* (Software Engineering Institute, Carnegie-Mellon University: Pittsburgh, Pennsylvania, January 1996), pp. 12-15. I should add that, *originally*, a "license" was a document someone would be

given (such as today's driver's license). Registration had a different administrative apparatus, a list or registry that could be checked if there was doubt about someone's status (such as today's voter registration roll). But this distinction is no longer sharp. Governments keep a record (a registry) of licensed drivers; and many jurisdictions issue a "registration card" (in effect, a license) entitling one to vote.

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

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

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

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

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

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

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

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

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

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

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

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

[26] Mary Shaw, "Prospects for an Engineering Discipline of Software", *IEEE Software* 7 (November 1990): 15-24. Selected runner-up to best article in *IEEE Software* for 1990, it had already been reprinted in Donald J. Reifer (ed), *Software Management*, 4th ed. (IEEE Press: New

York, 1993), pp.486-495). It was soon to be reprinted in *Encyclopedia of Software Engineering*, v. II, pp. 930-940.

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

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

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

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

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

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

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

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

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

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

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

# Chapter 3: SEEPP Begins, 1994

"Nothing is as easy as it looks."
—The First of Murphy's Laws

## 3.1 The Steering Committee

By March 4, 1994, the time at which the IEEE-CS Steering Committee for the Establishment of Software Engineering as a Profession presented its final report to the IEEE-CS Board of Governors, the new IEEE-CS/ACM Joint Steering Committee had already been working for at least three months. Indeed, the handing off of work from one committee to the other was so smooth that there does not seem to be a day on which it can be said to have occurred. We can be sure that it was complete by March 4 because Mario Barbacci, chair of both committees, had presented the first report of the new Joint Steering Committee to the ACM a few days before (in Nashville, Tennessee). By then, he had a new vice-chair from the ACM, Dennis Frailey, [1] and a much enlarged committee, all members but one identified in the report with an "A" (for ACM) or "C" (for IEEE-CS). The one exception, Patricia Douglas, member ex officio because she chaired the body-of-knowledge task force, was a psychologist at IBM Skill Dyamics. Though in fact a member of IEEE, she seems to have been treated as neutral. One or both of the following considerations explain her position: first, she seems to have had the trust of both the ACM and IEEE-CS; second, there was no one at the ACM (or anyone else at IEEE-CS) who wanted to co-chair her committee. Defining the body of knowledge was a daunting task best left to experts.

Six of the seven IEEE-CS members should be familiar from Chapter 1. Barbacci himself, Buckley, Chikofsky, and Druffel were from the original committee. Two of the ex officio members, Carver (Curriculum) and Melford (SEEPP), were from old IEEE-CS task forces (now IEEE-CS/ACM task forces). The one new IEEE-CS face, an "ex officio", Ron Hoelzeman, is an academic (Electrical Engineering-Computer Engineering, University of Pittsburgh), apparently a replacement for Schneidewind.[2]

Three of the seven ACM members are also familiar from Chapter 2.3. Zweben had served as the ACM representative to the old IEEE-CS Steering Committee. John Werth,[3] the new co-chair of the curriculum task force, had been (and was still) one of the Vice Chairs for Education of the *IEEE* Technical Council on Software Engineering (TCSE).[4] Gotterbarn, now co-chair of the ethics task force, had been active in earlier IEEE-CS discussions of licensing.[5] Mary Shaw is familiar for another reason, her extended critique of Buckley's original proposal at the TCSE meeting (2.4). The two other ACM appointees—Barry Boehm and Stuart Feldman—are new. Both would have carried considerable authority with other ACM members.

Barry Boehm holds a B.A. (Harvard, 1957), M.A. (UCLA, 1961), and Ph.D. (UCLA, 1964), all in math. He had worked at TRW from 1973 to 1989, ending his career there as Chief Scientist of the Defense Systems Group. Between 1989 and 1992, he was at the U.S. Department of Defense, serving as Director of the Information Science and Technology Office, Defense Advanced Research Projects Agency (DARPA), and as Director of the Software and Computer Technology Office, Defense Department Research and Engineering (DDR&E). Since autumn 1992, he had been TRW Professor of Software Engineering, Computer Science Department,

University of Southern California. He had authored several books, including *Software Engineering Economics* (Prentice Hall, 1982), *Controlling Software Projects: Management, Measurement, and Estimates* (Prentice Hall, 1986), and *Software Risk Management* (IEEE Computer Society Press, 1990), all then still in print.[6]

Feldman was at Bell Communications Research, Inc. ("Bellcore", now Telcordia Technologies) in Livingston, New Jersey. He held a management position in software engineering and computing systems and was the Technical Leader of the Telecommunications Information Networking Architecture Consortium, an international research group of telecommunications and computing companies. Feldman held an A.B. in Astrophysical Sciences and Mathematics (Princeton, 1968) and a Ph.D. in Applied Mathematics (MIT, 1973).[7]

As we look over this list (ignoring Douglas), we can see an obvious pattern. The Steering Committee is joint in a more complicated way than the official rationale admits—and divided in a way never officially stated. On the one hand, all the ACM members are (as a matter of fact) also members of IEEE-CS. This is not as surprising as it may seem. The membership overlap was (and remains) considerable. Some estimates are that 90% of ACM members are also members (or affiliates) of IEEE-CS (with only a somewhat smaller percentage running the other direction because the IEEE-CS is somewhat larger).[8] What we have, then, is not a distinction between IEEE-CS members of the Joint Steering Committee and ACM members but merely between IEEE-CS *appointees* and ACM *appointees*. What explains the overlap? Or, to put the question another way, what explains the continuing existence of *both* ACM *and* IEEE-CS? Why have they not long since merged?

The overlapping membership seems to have a relatively pedestrian explanation. Neither IEEE-CS nor ACM is a professional society (strictly so called) but a scientific or technical society, that is, an organization (formally) held together by interest in a field of knowledge (rather than, as a professional society would be, by occupation). To join IEEE-CS, one has to meet only *one* of the three criteria: a serious interest in any aspect of the computer field, membership in an affiliate society, or membership in IEEE or another IEEE society. Of the *three*, the hardest to satisfy may seem to be IEEE membership. But even that was not (and is not) hard for most ACM members. To join IEEE (as a "member", as opposed to an "affiliate")[9], one only needed a bachelor's degree from an ABET-accredited program *or a CSAB-accredited program (*or, absent such a degree, what the admissions committee would count as three years of experience in a position normally requiring such a bachelor's degree).[10] To join the ACM, one needed only a bachelor's degree (in any field), the equivalent level of education, or two years full-time employment in information technology.[11] Most members of ACM would have had more than two years full-time employment in information technology (in a position normally requiring a bachelor's degree), making them eligible for membership in IEEE (and so in IEEE-CS).

Why are there two organizations rather than one? The answer seems to be that, on the whole, they do different things. IEEE-CS shares with the rest of the IEEE a set of practical concerns characteristic of engineering, especially development of technical standards. The ACM, in contrast, is more concerned with the science of computing (especially, the mathematics of computing) and with the computer science curriculum. The two associations hold different sorts of conferences, publish different sorts of journals, and otherwise work side by side rather than in direct competition. *Collected Algorithms* or *Transactions of Programming Languages and*

*Systems is a* typical ACM journal; the IEEE-CS equivalent would be *Computer* or *Transactions on Mobile Computing*.

The explanation of this cohabitation is probably history rather than logic. The two associations began about the same time, one as a special interest group within an engineering society and the other as an independent society concerned more with the *applications* of computing machinery (what we now call "programs" or "software") than with the design of the machines themselves (engineering). As software became an ever larger part of the "machine", the distinction between engineering and applications became ever harder to make. Indeed, the term "software engineering" may be understood as a declaration that applications were as much in need of engineering as the hardware was. Nonetheless, there remain significant differences between the two organizations. Perhaps the most important is that the ACM has always had a stronger identification with student organizations in departments of computer science than IEEE-CS has.

### 3.2 The Odd Couple: Melford and Gotterbarn

These differences between the two societies may explain an important difference between ACM appointees and IEEE-CS appointees. All but one of the IEEE-CS appointees (Melford) have at least one (ABET-accredited) degree in engineering (or the non-US equivalent). None of the ACM appointees does. In contrast, all but one of the ACM appointees (Gotterbarn) have at least one degree in mathematics. The exceptions confirm what we know about the flexibility of the formal membership requirements of the two organizations.

Melford, a member of both organizations, was not an engineer; indeed, he was not even a college graduate. (When he left California State University at Irvine two years before graduation, his major was economics, though he had planned to study electrical engineering.)[12] He joined the IEEE in the early 1980s because a colleague encouraged him to. (By then, he had three years experience in security applications of software that would ordinarily be done by people with a bachelor's degree in computer science or engineering; hence, there was no problem about his membership even though he was not trained as an engineer.) Some years later he got a broadcast mailing from the IEEE-CS saying that the Committee on Public Policy (COPP) had started an ethics subcommittee and asking if he was interested in joining it. Because he had concluded that most problems with software were ultimately problems with those who wrote software, he joined. By the time the Steering Committee was setting up the task forces, Melford was COPP's chair, an obvious choice for IEEE-CS co-chair of the ethics task force.[13]

Donald Gotterbarn, the ACM co-chair, differs from Melford in at least three important ways. First is education. Gotterbarn had a lot more of it (and what he had was quite different from Melford's). Gotterbarn holds a B.A. in philosophy from Hofstra University (1964); a Master of Divinity from Colgate Rochester Divinity School (1967); and an M.A. (1970) and Ph.D. (1971), both in philosophy, from the University of Rochester. (His dissertation's title is *Natural and Philosophical Relations in David Hume*.) He taught philosophy at Wichita State University (Kansas) for three years (1971-74), turning down early tenure to accept an untenured assistant professorship at the University of Southern California's School of Philosophy. From the perspective of most philosophers, to teach at WSU is to toil in the lower reaches of academy's Purgatory; to move from there to USC, to rocket into its Heaven. So, what happened two years

later should surprise. In 1976, Gotterbarn left USC for Harrisburg, Pennsylvania, without even a job in hand.

There were several reasons for leaving USC just two years after he arrived. Both Gotterbarn and his wife were Easterners much farther from relatives than they had been in Kansas. Though they had grown up in New York City, they had developed a strong dislike for Los Angeles. These reasons, and perhaps others, led to the sudden re-direction of Gotterbarn's career.

For the Spring Term of 1977 and again for the 1979-80 academic year, Gotterbarn taught philosophy part time at Dickenson College (in nearby Carlisle), more because he liked teaching than because he needed the money. He made his living as a programmer.

Gotterbarn's preparation for programming was similar to Melford's, that is, he was (more or less) self-taught.[14] Having learned symbolic logic as part of his training in philosophy, he supplemented what he could teach himself from books on programming by sitting in on computer science courses at WSU and USC. To improve his job prospects in Harrisburg, he entered a two-year program in computing at Harrisburg Area Community College, graduating with an Associate of Arts (A.A.) in computing in 1979. (At the time, the A.A. was a recognized credential in applied computing.)

A second difference between Melford's background and Gotterbarn's is their respective careers within computing. Gotterbarn's early computing career is similar to Melford's (and was roughly contemporaneous with it). From 1976 through 1984, Gotterbarn worked as a programmer, moving quickly from simple programming to systems development. By the early 1980s, he had his own consulting firm. As a consultant, he did work on Medicare programs for Blue Cross Pennsylvania, helped Blue Cross develop networks between several of its regional operations, and developed inventory tracking systems for the US government's Ship Parts Control Center in nearby Mechanicsburg and a similar facility in Philadelphia.

But, for all this practical experience, Gotterbarn seems to have remained an academic, While there is a gap of fifteen years in the *Philosopher's Index* between his last publication concerned with Hume (1976) and his first publication concerned with computer ethics (1991), those years were not without publications. Gotterbarn's curriculum vitae lists (in addition to what the *Philosopher's Index* has for these years) the following: "James' Theory of Cognitive Knowledge", *Bicentennial Symposium* (1976);"Tipton on Berkeley's Immaterialism", *Philosophia* (1979); and nine entries for the *American Academic Encyclopedia* (1980).[15]

For the first few years after leaving USC, Gotterbarn seems to have tried to keep up with developments in his old field, the history of modern philosophy. But, more important, he began to identify ethical questions in the programming he was doing. The first such question concerned the transfer of some medical decisions until recently made by physicians, nurses, and other "health care providers" with direct knowledge of the patient, to computers or to distant Blue Cross employees with a computer assistant (an expert system). What responsibilities did programmers who designed or built the necessary software have as a result of this transfer of decision-making to people unable to see the consequences of their own decisions or to evaluate the assumptions on which the software was constructed? Did the programmers have the same responsibilities to patients as physicians did? Or greater responsibilities because their software had a much larger impact on patients than the decisions of any single physician could have?

Gotterbarn had also begun to work part time as a "faculty tutor" in the computer science lab at Harrisburg Area Community College.[16] Some of his students might (he realized) soon be

helping to design and build nuclear weapons. Was it right to teach them the programming skills they would need to do that? Then, in 1983, President Reagan announced the Space Defense Initiative (a space-based defense against nuclear missile attack heavily dependent on software of a scale, complexity, and reliability well beyond anything yet attempted). Here was another set of questions to consider. Many programmers doubted anything like the specified system could be built for several decades, if at all. They were also concerned that any system much like that proposed could only be trusted if it could be adequately tested under realistic conditions and such testing was not practical. (Who would want nuclear explosions going off a hundred miles or so above their heads?) Should programmers nonetheless work on such a project, since the technical problems were interesting and the money was good? What responsibility did programmers have to tell the truth about what they did, to warn clients and public about the risks inherent in a program too complex for anyone to understand, about the limits of what could be accomplished?

The similarity between Gotterbarn's career and Melford's ends here. By 1983, Gotterbarn was tired of consulting. (For him, too much of consulting was winning the contracts to do the work; his interest was the work itself.) So, in early 1984, he took a salaried position at a Scranton bank, designing electronic fund-transfer systems. (Scranton is about 120 miles north of Harrisburg.) When, a few months later, the bank was taken over by a larger bank uninterested in computer research, Gotterbarn turned down its offer of a position managing programmers to become a (much-lower-paid) assistant professor of computer science at Alleghany College (in Meadville, Pennsylvania, almost 300 miles west of Harrisburg). There he taught basic computer science courses and developed some new courses. Most of the new courses, such as one in database design, were technical. But one, a course in computer ethics, combined the technical with the ethical. Students actually developed software and reviewed technical documents looking for technical as well as ethical errors.

Gotterbarn was particularly well-suited to teach computer ethics. For many professors of computer science then, a course in computer ethics was not a computer science course at all. For them, computer science was a kind of mathematics, very far from practice. Computer ethics was a kind of ethics. Philosophers taught ethics. Gotterbarn's Ph.D. in philosophy thus seemed to make him the ideal person to teach the course. Gotterbarn also thought himself the ideal person to teach the course, but not (primarily) because of his training in philosophy. Unlike his colleagues, he had done programming "in the real world". He had, he liked to say, written programs that, written incorrectly, could kill people. He could convey to his students the uncomfortable truth that programming was not mere mathematics but a practical activity capable of doing great harm as well as great good. Part of avoiding harm was writing programs that met certain standards, standards of ethical conduct as well as of competent programming. Indeed, competent program was an ethical imperative (as well as a technical diseratum).

Courses in computer ethics were still rare enough then to be news. In 1988, the *Chronicle of Higher Education* carried a story entitled "Failure of Colleges to Teach Computer Ethics is Called an Oversight with Potentially Catastrophic Consequences". Gotterbarn was quoted in the article—along with another philosopher, Deborah Johnson (whose textbook *Computer Ethics* was described as one of the few in the field).[17]

During the years in Harrisburg, Gotterbarn had not had time to put his thoughts about computer ethics on paper. He had been doing too much else. Now, an academic again, he began to write—and to read what he was writing at conferences. The first of what would soon be many

papers was “The Roles of Computer Ethics in the Curriculum”, presented at the Conference on Computing and Undergraduate Education held at Carnegie Mellon University (1987).

Meadville is only 90 miles north of Pittsburgh where (as explained in Chapter 2.1) the Software Engineering Institute (SEI) opened on January 1, 1985. Gotterbarn was soon attending SEI seminars and conferences. Some of SEI’s first initiatives were educational. These included an attempt both to define a curriculum for software engineering and to raise the technical competence of software engineers already in practice. These were major undertakings. (At the time, as Gotterbarn recalls, the chair of his computer science department liked to say, “Software engineering is just about documentation.”). Gotterbarn approved of both initiatives. By late 1987, he was inquiring about participation in some of SEI’s educational initiatives.

In 1988, Gotterbarn went back to Wichita State, becoming an assistant professor of computer science. The move is not as surprising at it may appear. WSU’s department was the first to experiment with a Masters of Software Engineering using a curriculum SEI was developing. WSU thus gave Gotterbarn the chance to cooperate with SEI in the development of courses, to test them, and to help SEI draw from the results educational materials that could be distributed to other universities trying to establish similar programs in software engineering.

Early in 1989, Gotterbarn presented a paper on ethics at SEI and lobbied SEI staff to include ethics in the new curriculum. The lobbying proved effective. Gotterbarn was named a Visiting Scientist at SEI for the summer and asked to develop educational materials to be used in software engineering classes, including some materials on ethics. It was then, Gotterbarn believes, that he coined the term “software engineering ethics” to describe the ethical responsibility of software developers in the practice of their craft.[18] In any case, he began the summer with the first Software Engineering Ethics Workshop. It was there that he met computer scientists, Keith Miller (then at the College of William and Mary in Virginia) and Joyce Currie Little (Towson University, Maryland), both of whom had undergraduate degrees in (math) education (as well as Ph.D.’s in computer science). He would later recruit both for SEEPP.

By fall 1989, Gotterbarn knew a good deal more about WSU’s computer science department than he had a year earlier. It was a less happy organization than it had seemed at first and, because of a new chair, less committed to software engineering. Gotterbarn therefore began looking for another job. He soon found that East Tennessee State University (ETSU) was looking for someone to teach software engineering courses and to start a master’s degree in software engineering following the SEI model.

ETSU had other attractions to compensate for its isolation in northeastern Tennessee. Johnson City, with a population of about 60,000, was the size that Gotterbarn had found he liked. The location, a long valley, with the blue-ridged foothills of the Great Smokey Mountains rising abruptly on either side, was at least as beautiful as anything he had seen in Pennsylvania—and quite unlike the plains of Kansas (or the smoggy hills of Los Angeles). The university was of medium size (about 10,000 students) with a spacious campus of grass, trees, and streams. Its buildings spread out as if land were plentiful and cheap. Most of the buildings were of red brick with white trim in one approximation or another of Georgian colonial, giving the place an eastern look. Gilbert Hall, one of the oldest buildings on campus, was where Gotterbarn would have an office on the second floor, along with the rest of Computer Science (Mathematics on the floor below). Almost a century old, with an impressive façade, Gilbert’s interior was nonetheless utilitarian and undistinguished. Of course, the chief attraction of ETSU’s computer science department was not its offices. ETSU’s department was “applications oriented”, indeed, so

applications oriented that it already had a course in computer ethics. Gotterbarn did not have to explain to that department why it was important to teach computer ethics—or that a course in computer ethics was a computer course, not a philosophy course. He arrived in the fall of 1990 as an associate professor, moved into a windowless office, and has taught there ever since—except for an occasional visiting professorship elsewhere.

It was also about this time that Gotterbarn became a "Distinguished National ACM Lecturer" on ethical issues in computing (as wells as on fiber optic computers and new developments in telecommunications).[19] Gotterbarn was developing a reputation among ACM members interested in computer ethics.

### 3.3 The AMC Code

A third difference between Melford and Gotterbarn is that Gotterbarn had had some experience in writing a code of ethics before joining SEEPP, while Melford had none. Gotterbarn had helped to write the present ACM code of ethics. In 1991, Ronald E. Anderson, a former president of the ACM Special Interest Group in Computers and Society (SIGCAS), received a grant from ACM to revise its Code of Professional Conduct, adopted in 1972.[20] This was the first official endorsement of a project Anderson had already been actively working on for at least a year. He had begun that work at an all-day symposium at the first Computers and Quality of Life Conference (1990). Anderson presented ideas derived from that meeting at other meetings.[21] Now, grant in hand, he recruited from attendees at the annual Computers and Society Conference (February 1991) a dozen people to produce a draft. Among those recruited, beside Gotterbarn, were two professors of computer science whom Gotterbarn would later recruit as well (Dianne Martin and Laurie Werth).[22]

Over the next five months, Anderson prepared a "rough draft"—which he distributed at the Conference on Computing and Human Values, sponsored by the National Science Foundation and hosted by the Research Center for Computing and Society (RCCS) at Southern Connecticut State University (New Haven), in August, 1991. He also sent a copy of this draft to the ACM Council (the ACM's governing body).The Council's reaction illustrates how careful a drafting committee should be about who sees its early work—or, at least, about how it is presented. The Council's reaction to Anderson's rough draft was not good. The primary reason (he soon learned) was that Council members supposed that the rough draft was the final draft. Once that misunderstanding was cleared up, there remained a number of substantive criticisms.

At the Conference on Computing and Human Values, Anderson found the people he needed (about fifty) and the free late afternoons and early evenings to rework the rough draft. He presented the rough draft along with the criticisms he had collected. By the end of the conference, he had a presentable "first draft".[23]

Anderson distributed this official draft to the ACM Council. The Council responded. Anderson's drafting committee then revised accordingly (working in part at least by email). He sent the Council the revised code and received further responses which he tried to incorporate as he had before. Anderson was also presenting the draft at various ACM-sponsored meetings, collating responses from his audiences. Eventually, Anderson had what he supposed a final draft.

The process of creating a code of ethics, especially winning adoption, is always "political" (that is, an activity requiring participants to work together voluntarily, winning by persuasion what cannot be compelled). Anything from objective considerations (whom the code

would help or hurt) to personal taste in language might decide the outcome. As the drafting committee came to appreciate who wanted what and why, it revised the code accordingly. So, for example, Imperative 4.1 now simply urges ACM members to "Be fair and take action not to discriminate". Originally, it included a short list of prohibited subjects of discrimination, but various special interest groups wanted to add their own favorite to the list. To avoid a long list, the drafting committee decided to make it clear that the list was not intended to be exhaustive. The committee did that by moving the entire list from the Imperative to the corresponding Guideline. The generality of the Imperative now captures all types of discrimination. The Guideline provides enough specificity ("race, sex, religion, age, disability, national origin, or other such similar factors").[24]

The draft that emerged from this process was published in *Communications of the ACM* (May 1992) with a ballot asking members to vote on each item in the code. The vote was favorable for all provisions—clearing the way for the final Council approval. Then, a few days before the Council's meeting, Anderson learned that some Council members might vote against the new code because they believed it would not be useful in decision-making. (The Council included some members who had helped to write the code this new one was to replace.) Anderson asked Gotterbarn and Deborah Johnson to help write a paper showing the new code's usefulness in ethical decision-making. (By then, Johnson's text in computer ethics was widely used.) The three quickly wrote the paper, "Using the ACM Code of Ethics in Decision Making", and presented it at a Council meeting the morning of October 16, 1992.[25]

Since the presentation went well, neither they nor Anderson stayed for the afternoon session at which the code was to be approved. That may have been a mistake. The Council made a number of revisions before coming to a final vote. One of those revisions was significant (weakening the clause providing for penalties for disobeying the code). To this day, Gotterbarn believes that the drafting committee could have prevented those revisions had it been there to explain why it had written the code the way it had. One lesson Gotterbarn took from that experience is that those who know most about a proposed code should be present at deliberations concerning it until final approval. One cannot expect those who have not worked on the drafts and participated in extended discussions about them to understand the final draft as well as those who did.[26] Another lesson he might have drawn from his experience with writing the ACM code was that even a large organization could write a code of ethics in eighteen months.

### 3.4 SEEPP Gets Started

Gotterbarn first met Barbacci on November 16, 1993.[27] Gotterbarn was at SEI on business unrelated to the Steering Committee's work. In the morning, he video-taped a graduate class on professional issues in software management. (By then, SEI had a substantial distance education program.) In the afternoon, he made a CD for SEI's "Just in time education program". (SEI could generate CDs quickly in response to hot topics, marketing them to industry.) Between these activities, over lunch, he met with Barbacci. Barbacci believes Gotterbarn arranged the meeting.[28] (Gotterbarn does not remember.) Its official purpose was to "discuss" the IEEE-CS/ACM joint project. Barbacci sketched the plan so far developed, gave Gotterbarn copies of the minutes of the relevant meetings of the IEEE-CS Board of Governors and ACM Council, and asked about his interests, experience, and opinions. Barbacci also asked whether Gotterbarn was interested in participating in the project. He answered that he was.

A few days after the meeting, Gotterbarn decided to put his thoughts in better order. The result was a three-page single-spaced memo commenting on specific passages in the documents he had received. To this he added a copy of a paper he had written on licensing and emailed the package to Barbacci. The gist of the paper was that licensing was not an end in itself but a way to enforce standards, especially, standards of skill. The gist of the memo was that neither the IEEE-CS nor the ACM seemed to have a good grasp of what goes into making a profession; they certainly did not agree with Buckley that software engineering should be a profession. Getting agreement on significant standards was worthwhile, but significant standards had to be detailed, more detailed than either the IEEE-CS or ACM document suggested. Gotterbarn seemed to be advocating (what he would later describe as) "standards of practice" rather than a "mere" code of ethics. Neither the memo nor the accompanying paper has much to say about ethics. Indeed, the memo ends with a postscript asking, "How does ICCP [Institute for Certification of Computer Professionals] fit into the effort to define skills?"[29] Nowhere in the memo does Gotterbarn express any idea that he might co-chair one of the task forces (though he does thank Mario for "giving me the opportunity to participate").

Nonetheless, Gotterbarn was soon offered the opportunity to co-chair the ethics task force; he accepted; and, on January 29, 1994, Barbacci met with him and Melford in Pittsburgh to get the task force started. The meeting took place in the same SEI seminar room in which Gotterbarn had held his first workshop on software engineering ethics (1989). There was an irony in this. By 1992, thinking about ethics had changed at SEI. Norman Gibbs, then the Director of SEI's Education Division, had told Gotterbarn, "Ethics is not relevant to software engineering" and, proving he meant what he said, announced that SEI would not be publishing the curriculum module on ethics Gotterbarn had prepared. Now, less than two years later, ethics was back at SEI (if only on a weekend and only because Barbacci had his office there).[30]

This was when Gotterbarn and Melford first met. There was a long agenda. The meeting went well. The three began by refining (or broadening) the task force's assignment. The task force had been told (two months before) to develop a "code of ethics". The January minutes (Melford's) announce that "Codes of Ethics [are] (not equal to) Standards of Conduct", distinguishing them in this way: A code of ethics merely "identifies conflicts". Standards of conduct provide: "a) Guidance to resolve conflicts [between users, developers, employers, regulators, and others], b) Standards of quality, and c) Responsibility for understanding the implications of one's work." This distinction seems to explain the name by which the minutes refer to the task force for the first time, something that seems to need explanation (in part, at least, because the Steering Committee had created a task force to write a "code of ethics" and said nothing about "a code of professional practice"). The three had to reject (or overlook) the obvious name for the task force, the "Software Engineering Ethics Task Force", with its pleasant acronym, "SEE", for "Software Engineering Ethics *and Professional Practices*", with the unpleasant hint of ooze in the acronym "SEEPP". The three were unhappy enough with "SEEPP" that they agreed to find a better name "later". But events preempted good intentions. SEEPP had to use its name publicly before a better one could be found—and then a better would have had to be bought at the price of some confusion.[31]

The term "professional practices" suggests standards of conduct, something more than just "ethics". In choosing to include "professional practices" in the task force name, the three may seem to have been committing SEEPP to report something more than a code of ethics, that is, a "code of practice". There is, however, another interpretation—for Gotterbarn, at least. The

three-page memo he had sent Barbacci after their first meeting (November 16) included the following comment (concerning "page two"):

> In general, I would suggest keeping constantly coupled the phrases "ethical conduct and professional practice." The definitions of professional you included do not indicate the contractual nature of a profession. The theoretical contract of professionalism says that "by virtue of having a specialized skill and being allowed by society to maintain control over this skill, society is due a HIGHER standard of care from the (licensed) professional. Ethical conduct is tied to professional practice.[32]

For Gotterbarn, then, the point of joining "ethical conduct" with "professional practice" is (in part at least) to stress that "professional practice" is "ethical conduct" (conduct made ethical by a moral "contract" between profession and society).

The commitment to something more than a code of ethics, if there was then such a commitment, may explain something else all three agreed to: adopting the *IEEE*'s "Standards-development model" (even though the *Joint* Steering Committee and its subcommittees belonged to the ACM as much as to the IEEE). For the IEEE, any Standard-developing committee (or other committee assigned a large project) becomes a "steering committee" that oversees the work of several "task forces", one task force for each major division of the project. Each task force consists of several "working group leaders", one leader (or occasionally more) for each subdivision of the project. Each working group leader oversees a specialized working group and reports back to the task force. The three agreed on a preliminary list of working groups: privacy; security; testing and reliability; professional competence; duties to society; intellectual property; and ethical tools.[33]

The three also agreed to follow IEEE procedures for writing "Standards" (the capital "S" signaling a certain sort of standard). For the IEEE a standard (with a small "s") is a general category consisting of "Standards" (with a capital "S"), "recommended practices" (marked by a "should" rather than the "shall" of a Standard), and "guides" (more than one practice can be recommended). Because Gotterbarn knew little about IEEE procedures, Melford undertook to prepare the required "Call for Participation" (CFP) the IEEE way. This is the first sign of a pattern that would endure for almost two years—with nearly disastrous consequences (as we shall see in subsequent chapters). Gotterbarn, the ACM's co-chair, would defer to Melford (and Barbacci) on SEEPP procedure because the IEEE seemed to have (and actually did have) a good deal of experience writing standards, much more experience than the ACM, and writing a code of ethics-*and-professional-practice* seemed to be much more like writing other IEEE standards (whether Standards strictly so called, guidelines, or mere recommended practices) than like writing a mere code of ethics.

The three did not, however, rely entirely on the Call to bring in participants in the numbers and variety they wished. They had already recruited three members for the task force: Joyce Currie Little, identified in the minutes as an "educator"; Keith Miller, as the "ACM ethicist"; and Gerald Engel (newly returned to teaching after several years on loan to the National Science Foundation), as the "Grant Funding Advisor".[34] Gotterbarn undertook to find a corporate ethics officer, and a lawyer with interests in malpractice, negligence, liability, and intellectual property. Melford undertook to contact some other "ethicists" (Patrick Sullivan and Michael McFarland); some experts in the IEEE's Standards process (Leonard Tripp and Fletcher

Buckley); and the chair of the IEEE-CS ethics subcommittee that Melford had himself once chaired (Suzanne Weisband). While there was agreement that the task force or its working groups should include "technologists", no one undertook to recruit any (perhaps because most of those already identified for other purposes fit that description).

Last, the three set a target date for completion of the work of the task force, November 1995. The task force would have twenty-two months, four months more than Anderson's committee needed to write the ACM code (and win its approval). The three sketched a tentative schedule, beginning with the next meeting ("#1") to be held on Saturday, April 30, 1994, and (after "meeting #4") ending with "Submit (Nov. 95) report and draft to [IEEE-CS] BOG".[35]

The agenda of the planned next meeting (April 30) was ambitious. The meeting was to:

1. Revise Task Force schedule,
2. Partition/categorization of standards,
3. Funding grants,
4. Establish deadlines for WG [working group] draft standards,
5. Complete email/conferencing/public bulletin board,
6. Formulate draft PAR [Project Authorization Request] (for internal use only) to identify components for successful submission,
7. Prepare call for participation (CFP) in working groups,
8. Identify audience and time limit for CFP (e.g. 30 days),
9. Identify WG chairs (continues).[36]

### 3.5 PAR in DC?

On April 24, Melford sent out an official agenda for the April 30 meeting; it summarized the January 29 meeting, refined the agenda items, and allotted time for each item (for example, "9:15-10:00, Define Scope of ethics TF [task force]; 10:00-10:15, Revise Task Force Schedule; 10:15-10:30, Library research: identify titles/documents, identify 'who gets what', and document repository"). The meeting was to end some time after 3:00 pm—after the task-force schedule had been "revised (again)".

The timing and location of the meeting deserve comment. The meeting was scheduled in Washington, DC so as to come just after the annual meeting of the Computer Ethics Institute (CEI). The CEI was a good place to advertise SEEPP—as well as an important capital in the geography of computer ethics. CEI began in the mid-1980s as the Coalition for Computer Ethics, a local discussion group. Support came mostly from IBM, the Brookings Institution, and the Washington Theological Consortium, as did most participants. In 1992, it formalized itself as CEI, a project of Brookings. Though similar in name to SEI, CEI was (and remains) a quite different entity, "an advanced forum and resource for identifying, assessing, and responding to ethical issues associated with the advancement of information technologies in society". CEI is supposed to stimulate awareness of ethical issues in technology by consultation, research, education, and public outreach, but *not* to set standards (though it did develop "The Ten Commandments of Computer Ethics"—a short code designed for all computer users—in the early 1990s). CEI's primary focus seems to be organizing events bringing together people interested in public policy issues related to computers (both hardware and software).[37]

Melford had been active in CEI from the late 1980s. That was how he knew Patrick Sullivan, its executive director since 1991. Sullivan was a philosopher (Ph.D. from the University of Kentucky, 1988, dissertation on Pierce's speculative rhetoric) who (like Gotterbarn) had developed an interest in computing. But (unlike Gotterbarn) Sullivan never became a programmer, much less a software engineer. Having followed his spouse to Washington in 1991, he combined some teaching (everything from introduction to philosophy at a small college nearby to the graduate bioethics continuing education program at Johns Hopkins). CEI was an ideal place for a philosopher to meet others with a sophisticated understanding of computer ethics.[38] Sullivan had met Gotterbarn at an earlier CEI conference. So, when Melford suggested that Gotterbarn be invited to participate in CEI's third annual Conference on Computer Ethics (*Building Ethical Culture in Cyberspace: Further Pursuit of a "Ten Commandments" for Computer Ethics)*, Sullivan was happy to arrange a panel at which Gotterbarn (along with Melford) could discuss software engineering ethics. (The invitation meant that ETSU could cover Gotterbarn's travel costs—important because SEEPP still had no budget.)

SEEPP's meeting was held, as planned, at the Washington headquarters of the IEEE-CS—but that seems to be the only thing that went as planned. No minutes survive but, in a memo written a year later, Gotterbarn described the meeting in much less ambitious terms than the agenda had. A "get acquainted meeting", it was a small affair, with only Gotterbarn, Melford, Steve Barber (a lawyer whom Gotterbarn had met a few weeks before)[39], Little, Sullivan, Ramon Barquin (president of CEI and a founder of the Coalition when he was at IBM), and Elliott Chikofsky (Steering Committee member who happened to be in town) physically present.[40] Buckley and Tripp participated by conference call. The discussion could not advance much beyond January 29 because the documents that such an advanced discussion presupposed (a draft of the Call for Participation, the PAR statement, and so on) were not ready. Gotterbarn recalls Melford asking Tripp detailed questions about how to prepare a PAR—with no one else present having any idea what they were talking about. Barber just remembers "a telephone conference with an IEEE process guru that seemed to go on forever with no real benefit. SEEPP members familiar with the process guru engaged in quite a bit of eye-rolling."[41]

### 3.6 Report to Steering Committee

On May 28, 1994, at the Software Engineering Institute, the Joint Steering Committee held its next official meeting (starting at 10:00 and ending at 5:00, with an hour for lunch). All Committee members were present except Buckley, Melford, and Shaw.[42] Each task force reported, beginning with the body of knowledge (Douglas) and ending with the curriculum (Carver and Werth). What each reported was a sketch of the work ahead. "Discussion" followed. The meeting's minutes use one page single-spaced to summarize the three reports but three-and-a-half pages to summarize the "Discussion". Of those three-and-a-half pages, about two thirds concern the body of knowledge, with the remainder divided evenly between curriculum and ethics. (The schedule of the meeting reveals a similar emphasis, with the body of knowledge receiving seventy-five minutes to sixty for ethics and forty-five for curriculum.) Clearly, for the Steering Committee, defining the body of knowledge was the most interesting, or at least the most troubling, part of their project. But our concern is SEEPP.

Gotterbarn spoke for his task force. All members of a profession have obligations because they belong to that profession. Reflecting on the obligations of other professions, the

task force identified three "orthogonal axes" by which other professions organize their special obligations. One is the *groups* to which they have special responsibilities (the individual, the organization, the society). A second axes is the *standards* from which the obligations derive (justice, equity, privacy, autonomy, beneficence, avoidance of harm, fidelity, integrity). The third axis is defined by *problems* (privacy, security, testing and reliability, competence, duties to society, intellectual property, and ethical tools).[43] The point, apparently, was that though SEEPP seemed so far to be working as if the third axis (the problems) were the only possible axis, it was aware of the other two. The other two were certainly potentially important. For example, the old IEEE code of ethics (1987) was organized by group, while the new one (1990) was organized by standard.

Gotterbarn also indicated that SEEPP intended to follow the IEEE Standards process the scribe "correcting" Gotterbarn's description with a bracketed "actually, ANSI" Standards process). While the minutes do not suggest that anyone on the Steering Committee had any doubts about using the IEEE Standards process, Gotterbarn recalls a few familiar with it (Druffel, Chikofsky, and perhaps someone else) privately recommending against following the whole elaborate process, because the process usually took "too long" (that is, longer than anyone wanted the work of the Steering Committee to take). Having had no experience with the IEEE process, Gotterbarn had no idea that "too long" meant "five to seven years" (the normal time for developing a standard). He was still learning his way around IEEE-CS.

The recorded discussion of Gotterbarn's report concerned the "target" of the standards SEEPP would develop. Most of that discussion (as the minutes report it) took the form of a four-layer chart. At the top is "User Programming (100 million people in US)". The next layer consists of categories of people: "Applications Generators (0.2 M), "Application Composition (2 M)", and "Systems Integrators (2 million)". Below these are "Infrastructure (1 million)"; and, at the bottom, "Academics Foundation Infrastructure (0.01 million)". The discussion concluded that the task force should target "the middle layer". This leaves the impression that the target is to be everyone between User Programming and Academics, more than five million people in the US alone, a group at least five times larger than any published estimate of the number of "software engineers". So, perhaps, the Infrastructure group was what was actually intended. (Gotterbarn believes that this chart was Boehm's contribution: "Boehm was always 'chunking' [that is, grouping ideas or making distinctions]."[44]

The only "action" the Steering Committee took relevant to SEEPP was to assign Barbacci and Zweben responsibility for a press release from the "TF on Ethics and Professional Practices through CS/ACM… at House briefing on June 21".[45] For those already cynical about the IEEE-CS and ACM undertaking to make software engineering a profession, this briefing will seem an important event. Participants not in the Washington area (including both Gotterbarn and Melford) had expenses paid by IEEE-CS or ACM. They were brought to a meeting room in the House's Rayburn Office Building and seated in a row facing a few reporters.[46] During a presentation that lasted only a few minutes, the speaker said something like "We are working on ethical issues in computing" and pointed to Gotterbarn and Melford. They had no opportunity to speak or answer questions. Gotterbarn had never seen the speaker before—and never saw him again. Soon their part of the briefing was over and the co-chairs were on their way out of town, Gotterbarn having no idea who had arranged the event or why.[47]

### 3.7 Defining the working groups

On June 11 (ten days before the briefing), the task force held its second scheduled meeting. The host was Michael McFarland, then a professor of computer science at Boston College.[48] The location was a local monastery. Of the eight members of the task force, six were physically present—with a seventh (Weisband) present by conference call. Only Barber could not attend even by speaker phone. The first half of the meeting consisted of a sorting out of research tasks: There was a need for an online library to hold codes of ethics (IEEE, ACM, British Computer Society, Australian Computer Society, and so on, including several European codes), bibliographies, course syllabi on computer ethics, conference programs, case studies, and so on. McFarland would find a piece on casuistry to include. A graduate student "from somewhere" would manage the library. We never again hear of this library.

There followed a discussion of how to circulate the (yet to be prepared) Call for Participation. The first venues discussed were all list servers or electronic bulletin boards, with paper publications (*Communications of the ACM*, *SIGCAS*, and so on) considered next. Melford (again) undertook to prepare a "general boilerplate" Call and circulate it to the working group chairs.

The eight working groups were tentatively set up, that is, a chair (drawn from the task force) was named for each. McFarland took on *Institutional Support*; Gotterbarn, *Professional Competence*; Weisband and Barber, *Intellectual Property*; Little, *Professional Relations*; Sullivan, *Privacy*: Miller, *Reliability and Safety*; Melford, *Security*; and Weisband, *Social Justice*.[49]

We might make three observations concerning these working groups . The first is that their titles (or "problems") have changed a bit since May. "Institutional support", "professional relations", and "social justice" have been added; "duties to society" and "ethical tools" have disappeared. Perhaps "social justice" is a mere synonym for "duties to society", but "ethical tools" does not seem shorthand for "institutional support" or "professional relations" (or any obvious combination of them).

The second observation is that understanding the architecture of this list of "problems" (or the one it replaces) is difficult. The list does not seem exhaustive. Some common issues of software engineering ethics, such as conflict of interest or truth in advocating a design option, do not seem to fit snugly under any of the eight categories. The list also seems not to consist of mutually exclusive categories. Some issues, such as protecting information or protecting end-users from harm, seem to fit under two or more of the eight categories (say, safety and social justice). The looseness of the architecture seems to have been noticed. Half the meeting's minutes consist of a "discussion of distinguishing content". So, for example, "Professional Competence" is analyzed as "avoid generalization of expertise, keep current, keep staff current, truth about skill, what should be produced, appropriate knowledge base". "Privacy" is distinguished from "Security" because privacy is "Nightline shows, Monitoring (workplace, everywhere), data aggregation, data valence—digital persona, misuse of data" while security is "illegal stuff, data integrity, trap doors, system administrator responsibility, malicious programming, encryption".

Clearly, the question of architecture remained. This suggests that we might better understand the division of labor among working groups by looking at who was assigned to chair the group. Since no one now recalls how this list came about, we must try to reconstruct what might have happened. The only obvious hypothesis is that there is a working group where there

is a potential chair already interested in the problem. We can find two sorts of evidence for this hypothesis (beyond the lack of obvious architecture).

The first consists of what we know about the interests of the working-group chairs. For some, the evidence is pretty clear. Melford made his living making computer systems secure. His interest in security went back to his undergraduate days.[50] Gotterbarn (as he made clear to Barbacci) thought of ethics as a way of helping to assure professional competence. McFarland had published two papers on computer ethics, each with a final section on structuring the institutions in which computer professionals work to help them do the right thing.[51] Long active in the Institute for Certification of Computer Professionals (ICCP), Little was interested in relations among professionals (which she understood as primarily professional organization).[52] Miller had published a half dozen articles (and had given a larger number of papers published in transactions) on testing and reliability in the four years just before joining SEEPP.[53] Sullivan's interest in privacy had been developing for several years and would lead him, only three years later, to leave CEI to join Price Waterhouse's new consulting group concerned with compliance with U.S. and international privacy and data protection regulations.[54] Weisband would have brought a concern for social justice with her from the Committee on Public Policy.[55]

Barber appears to be the chief exception. He was recruited because he specialized in intellectual property; yet he is not chair of the working group on intellectual property; Suzanne Weisband is. Weisband was first named as a possible participant in the organizing meeting on January 29 without any subject being mentioned. But she had recently published two (short) papers on software piracy.[56] Perhaps she reminded Melford of that when he contacted her in February to invite her to SEEPP's April meeting. She is listed ahead of Barber under Intellectual Property, out of alphabetical order, the normal sign of serving as chair. Neither Barber's interest nor legal credential was enough to get him his own chair. Apparently, Weisband got the chair because she outranked him (or because he was absent); Barber had to settle for vice chair. Thus Barber is one of those exceptions that, while not necessarily proving the rule, at least does not disprove it.

The working group chairs are, admittedly, knowledgeable about the problems assigned them as well as interested. But their knowledge does not explain the list. Knowledge (or competence) would explain why they were appointed as chairs once the list had been drawn up but something else must explain the list. On the one hand, knowledge did not guarantee that one would be the chair. Barber doubtless knew more about intellectual property than Weisband and she had another chairmanship (Social Justice). She nonetheless took the chair of Intellectual Property, leaving Barber to be vice chair. On the other hand, the working-group chairs doubtless knew much about many problems not on the list, such as continuing education or enforcement of standards. They must also have thought many other problems important enough for SEEPP to consider; they had, after all, earlier produced a somewhat different list of problems for the working groups. But if no one expressed an interest in some of the original problems, those problems would disappear from the list.

The second sort of evidence for the hypothesis that the list of groups mirrors the interests of participants rather than any plan for a code of ethics is in the Call for Participation. The Call (in all published versions) speaks of "working groups…being formed to address specific areas of *concern*" (my italics). Though eight are then identified, the Call adds, "Should additional subject areas be identified, additional working groups may be established."

## 3.8 The Call goes out

On August 24, Melford faxed Gotterbarn a draft of the Call for Participation. A week later, Gotterbarn sent all working group leaders the draft, asking for comments. Having received almost no comments by the end of October, Gotterbarn made a few changes and sent out the revised draft for "final" comment by November 5, 1994. As soon as the deadline for comment had passed, he sent out the announcement to all the publications previously identified as appropriate venues for the Call. The Call was a substantial document, four pages single-spaced (with a two-page application form appended). While the official Call did not go out until November, Melford seems to have begun sending out (short) unofficial versions earlier (at least as early as June)—without telling Gotterbarn.[57]

The Call (November 5) begins "Dear Computer Professional". It had six main divisions (as well as an untitled introductory paragraph explaining the larger IEEE-CS/ACM undertaking). The main divisions are: Scope, Purpose, Organization, Participation, Internet Access, Completion, and Thank You. Each of these main divisions is numbered, beginning with "1.0" for Scope and ending with "6.0" for Thank You. Most divisions are quite short (a single sentence for all but Scope and Organization). But Organization, with eight subdivisions, covers two-and-half pages.

The Scope 1.0 states that the task force would "document generally accepted principles for identifying and resolving ethical conflicts" in software engineering—suggesting a (descriptive) survey rather than a (prescriptive) codification. The Scope then provides a list of stakeholders (persons or groups whom the software engineer might have special responsibilities or obligations to: "peers and laypersons, employer, customer(s), the profession, and society/humanity". But the Scope also includes in the task force's work "consideration… [of] the obligations and responsibilities of these various entities towards the Software Engineer" (something the Purpose statement immediately following omits). This reference to obligations of those who are not software engineers suggests a document (like the American Medical Association's 1847 code of ethics) that states *reciprocal* obligations between an occupation and other stakeholders ("rights and responsibilities"), not—like a typical professional code—a simple statement of the profession's obligations. We never again hear of the obligations of others to software engineers.

Scope 1.1 adds that the task force recognizes that the profession "transcends national boundaries". Any standards the task force developed should "be as reflective of the global computing community's wisdom as can be reasonably achieved." The standard will state what is generally accepted or, where no one standard is generally accepted, "various recommended practices and guidelines".

Organization 3.0 describes the eight working groups in more detail than the June minutes, but in much the same terms—except that Barber is now explicitly designated as vice chair of the working group on intellectual property (with Weisband as chair).

Participation, the first subsection under Organization, contains several surprises. 3.1 declares participation "open to all individuals who are directly or materially affected or interested in" any issue within the Scope of the working group. "Members of the international computing community" are "particularly encouraged to participate". Essentially, *anyone* who

wants to participate can, presumably even a gaucho on the Argentine plains or an Imam in the holy city of Qom. Participation 3.1.1 adds that individuals may participate in as many working groups as they desire. 3.1.2 allows organizations to send representatives but warns that no organization will have a "veto". After all these inviting words, 3.1.3—all in caps—comes as a surprise: "PLEASE COMPLETE THE APPLICATION AT THE END OF THIS DOCUMENT AND E-MAIL IT TO SEEPP—CFP@COMPUTER.ORG, BEFORE 5-NOVEMBER-1994. We are organizing the working groups on 12 November 1994." The November 12 deadline—at most, a week from the date on which the Call was issued—is reasonable if, but only if, the expectation was that great numbers of potential participants were ready to apply and most of them would read the Call soon after it was sent out. If, however, filling out the form was demanding enough to be put off for a time for other things, or if many who might read the message would wait till it had been printed and distributed, or if the pool of potential participants was in fact small, any deadline, but especially one so tight, seems to be a mistake. It is a good sign, then, that when Gotterbarn and Melford met again, along with Sullivan, on November 11, at a computer ethics conference in Washington, DC, they already had two dozen or so applications to look at.

Later versions of the Call, such as that printed in the December 1994 issue of *Computers and Society*, never again contained a deadline for application. Earlier versions of the Call—for example, the preliminary October version that Gotterbarn had sent out to some lists in October[58]—also contained no deadline. The November 5 deadline thus seems to have been a (temporary) attempt to speed the process rather than a device to keep the number of participants down—and it seems to have worked.

### 3.9 A good report

A few days after the November meeting, Gotterbarn sent the list of those responding to the Call to all the working group leaders, asking them to identify anyone interested in their working group, organize mailing lists for their group, and get to work. By then, the applicant pool was over thirty (with many indicating they wanted to participate in two or more of the working groups).

So, when, on November 18, the Steering Committee had to report to the IEEE-CS Board of Governors on its activities for the year, it could describe SEEPP's achievements with at least as much pride as those of the other two task forces:

> Meetings
> - April 30, 1994, CS Headquarters
> - June 11, 1994, Boston College
> - November 11, 1994, Ethics in the Computer Age Conference

> Working groups
> - Institutional support
> - Intellectual property
> - Professional competence
> - Professional relationships
> - Privacy
> - Reliability and safety

- Security
- Social justice

Call for participation in wide circulation

These achievements were apparently substantial enough that, in December, Barbacci undertook to cover the task force's overhead (a few conference calls and some travel).[59] The payment of these expenses seems significant only because the Call makes clear (Finances 3.1.3) that most expenses will not be covered: "Participation is voluntary. Each individual is responsible for expenses incurred to support his/her participation. Many institutions provide support for their volunteers."

NOTES

[1] Dennis Frailey has BS in mathematics from Notre Dame University (1960) and an MS and PhD in computer science from Purdue (1971). He claims to have worked in software engineering before there was such a field (that is, since 1962). He worked for Texas Instruments and Ratheon while also serving as an adjunct professor at Southern Methodist University. See http://engr.smu.edu/~frailey/bio.html (August 28, 2004). Frailey became vice chair when Zweben, having become AMC President, became a a mere *ex officio* member.

[2] While Hoelzeman is listed as a member ex officio, that designation seems to be a mistake. The "ex officio" are task force chairs. Hoelzeman is not a task-force chair (and seems to hold no other office that would entitle him to that designation—though he was then a member of the IEEE-CS Board of Governors). He can only be Schneidewind's replacement.

[3] John Werth has a BS and MS in Mathematics, Emory University (1962 and 1963) and a PhD in Mathematics from University of Washington (1968). He was at this time Senior Lecturer and Research Scientist, Computer Science Department, University of Texas, Austin. http: //www.cs.utexas.edu/users/jwerth (August 28, 2004).

[4] Laurie H. Werth (John Worth's spouse), was an early SEEPP volunteer. She too taught computer science at the UT-Austin. Gotterbarn\SEEP1994-95, sec. 11.

[5] See Chapter 2.3.

[6] http://sunset.usc.edu/Research_Group/barry.html (August 28, 2004). I have corrected some dates of publication.

[7] Feldman declined to be interviewed for this book, claiming to have had nothing to do with SEEPP, even though his own biography lists him as a member of the Joint ACM-IEEE task force. See http://www.research.ibm.com/iac/advisory-feldman.html (August 28, 2004). Chapter 8.6 describes Feldman's part in writing a 1997 disclaimer that SEEPP was to put in published versions of the code stating that the publication of the Code draft (Version 3) did not indicate any approval of the Steering Committee. Perhaps this was the extent of Feldman's participation and, from his perspective, not a significant contribution to the code.

[8] I have not been able to find a written source for these estimates.

[9] An affiliate member of IEEE-CS has vote in the CS but not in its parent IEEE.

[10] http://www.ieee.org/membership (August 28, 2004). Since 2001, ABET and CSAB have worked together on accreditation, dispatching joint committees.

[11] http://www.acm.org/membership (August 28, 2004).

[12] That is, he did not receive a degree from an ABET-accredited engineering program and probably could not be licensed as a P.E. (since he lacks an appropriate degree). Or, as Melford himself put it, in answer to the question, "Are you an engineer?" (Interview of Melford, November 1, 2002): "Not officially. I'm not licensed and I don't have a degree. But most people would consider me to be an engineer. I think most of my colleagues view me as such. I certainly identify with that side of my field, probably more than anything else other than general business. I am an engineer in the sense that I determine how technology can be applied in a cohesive and complimentary manner to solve problems." See also his answer to question 1, about his education (endnote 15 below).

[13] Interview of Melford, November 1, 2002.

[14] "I left school to earn a living. I was studying economics. I have some formal education in economics and some in programming. I am fully self-educated in systems management as well as the business side of computing operations. I am fully self-educated in networking, the technical aspects of networking, as well as the ethics. One of my incentives for leaving school was that I wasn't able to get what I was after. At the time, there was basically nothing available on computing security. I originally went in to study electrical engineering, but found that that wasn't quite my forte. I came into computing in a backward sort of way. I read some issues of *Scientific American* in the late '70s or early '80s. I believe it was the September 1980 issue that was devoted to automation in the workplace, and the following issue in October was dedicated to software computing. Those two articles were extremely influential on me. They made me think about automating the business workplace. During that time, I had taken several courses in programming and management of information systems (MIS). I found that I had a knack for it. I also took programming classes that were oriented toward business majors, towards accounting students. That was at Cal State Northridge. When I left school I decided that I wanted to continue my education on my own. So I went and bought a PDD11. Yes, that was the computer Digital Equipment Corporation then made, the old deck PDD11, forerunner of the Vax. I mounted an early version of UNIX on it, and started to go from there. That led into some consultant work. I started my own consulting practice in 1984. I did consulting for 15-16 years." Interview of Melford, November 1, 2002.

[15] Gotterbarn's vitae also lists (among philosophy publications during this period): "A Model for Software Engineering Ethics" in *Lecture Notes in Computer Science: Software Engineering Education* (Springer Verlag, 1989). Amid this philosophy, are Gotterbarn's first

publications in computer science: “An Optical Bus Architecture”, Technical Report F86-2, *Allegany College Technical Report Series* (1986): and a contribution to pedagogy, *A Study Guide for Hexadecimal Arithmetic* (Harrisburg Area Community College, 1980).

[16] Gotterbarn could be a “faculty tutor” because he was also teaching philosophy at HACC (Introduction to Philosophy and Logic).

[17] Thomas J. Deloughry, “Failure of Colleges to Teach Computer Ethics is Called a Oversight with Potentially Catastrophic Consequences”, *Chronicle of Higher Education, February 24,* 1988.

[18] Gotterbarn later wrote the first encyclopedia article on software engineering ethics for the *Encyclopedia of Software Engineering* (1994), p. 1197-1200.

[19] The ACM pays the airfare of a Lecturer for a visit to an ACM professional or student chapter. The local organization covers food, hotel, and other local expenses.

[20] Anderson is (and was then) a sociologist at University of Minnesota (Ph.D., Stanford, 1970). For an explanation of the code—and a brief history of writing a code much less messy than the one described here—, see Ronald E. Anderson, “The ACM Code of Ethics: History, Process, and Implications”, in Chuck Huff and Thomas Finhold, editors, *Social Issues in Computing: Putting Computing in its Place* (McGraw-Hill, Inc.: New York, 1994), pp. 48-62.

[21] The most important of these conferences were: the Computer Science Conference and the National Educational Computer Conference.

[22] After Anderson, the ACM code lists as authors (with the parenthetical affiliations being my addition): Gerald Engel (University of Connecticut-Stamford, but then on loan to NSF), Donald Gotterbarn (ETSU), Grace C. Hertlein (California State-Chico), Alex Hoffman (Texas Christian University), Bruce Jawer (IBM), Deborah G. Johnson (Rensselaer Polytechnic Institute), Doris K. Lidtke (Towson State), Joyce Currie Little (Towson State), Dianne Martin (George Washington), Donn B. Parker (Standards Research Institute, but with an early book on computer ethics), Judith A. Perrolle (Northeastern University), Richard S. Rosenberg (University of British Columbia). Note that all but two (that is, 4/5’s—*not* counting Anderson) are academics. Anderson, “ACM Code History”, does not explain how these few names were chosen from the larger number of those who participated in some way or another in writing the ACM code.

[23] Among others whom Gotterbarn first met at the conference and would later recruit for SEEPP was a philosopher at the RCCS (John Fodor, who had only recently received a Ph.D. from Washington University-St. Louis).

[24] Donald Gotterbarn, “Two Computer-Related Codes”, *Perspectives on the Professions* 19 (Fall 1999): 5-6

[25] This paper eventually appeared in the *Communications of the ACM* (February 1993): 98-108. Judith Perrolle also listed as authors. This paper and presentation were Johnson's only part in "writing" the ACM code of ethics. Apparently, one can become an author by lending one's authority to a text (authorizing it) as well as by helping to write it (authoring it).

[26] Gotterbarn, "Two Computer-Related Codes".

[27] X (memo's name) gives the date as "November 10". That seems to be a typo.

[28] Barbacci, email, September 12, 2005.

[29] Undated Gotterbarn memo beginning "Mario, I enjoyed meeting". (paper only).

[30] Gotterbarn's emailed comments on March 24, 2003 draft of this chapter, p. 9.

[31] Gotterbarn, email, April 24, 2003.

[32] Undated Gotterbarn memo beginning "Mario, I enjoyed meeting" (paper only).

[33] Anderson had never divided work on the ACM code into parts. His group had always treated the code as a single indivisible undertaking. Hence, the decision to divide SEEPP into working groups does not seem likely to have originated with Gotterbarn.

[34] Gotterbarn knew Engel from ACM (see below). Barbacci and Melford also knew him—from IEEE-CS.

[35] Gotterbarn\ SEEP1994-95\FIRST meeting 1-94.

[36] Gotterbarn\SEEPP1994-95\SEEP2A.

[37] http://www.brook.edu/its/cei/cei_hp.htm (August 28, 2004).

[39] This was at the fourth "Conference on Computers, Freedom, and Privacy", held at the John Marshall Law School, Chicago, March 23-26, 1995.Gotterbarn, who had organized the panel ("The Licensing of Computer Professionals"), is identified in the panel's abstract as "chair of the SIGCAS Task Force on Licensing and Certification of Computing Professions". Barber was on the panel (more or less) by accident. Gotterbarn had initially asked Lance Rose, a New York City attorney specializing in software and online intellectual property. Rose had previously spoken on licensing to a software consultants' group in New Jersey. Since Rose was doing other things at the conference and felt he did not have much more to say on the panel's subject, he suggested one of his associates, Barber, instead. Barber, though new to lawyering (J.D., Yeshiva University 1993), seemed a good choice. He had a B.S. in Computer Science and Engineering (MIT, 1987). From 1984-1990, he had worked as an in-house developer for a vendor of packaged commercial software. He had worked on IBM protocol emulation packages for PCs and Unix-based systems. He was exposed to and performed tasks related to almost all phases of

the software development lifecycle—module design, implementation, testing, in-house and field maintenance, and project management. He also co-wrote and edited a technical book (Lewis, Barber, and Siegel, *Programming in Java IDL*, Wiley 1987). Gotterbarn, impressed by what he heard (arguments *against* professionalization), invited Barber to dinner, laid out his plans for SEEPP, and invited Barber to join. Interview of Barber, November 14, 2002.

[40] Gotterbarn\History of SE Code\History expanded, under 4/95 meeting: "partial list of attendees. Robert Melford, Don Gotterbarn, Steve Barber, Joyce Currie Little, Patrick Sullivan, and Fletcher Buckley and Suzie Weisband on conference call. [where did I find out about Chikofsky and McFarland?]

[41] Interview of Barber, November 14, 2002.

[42] There was also a "guest" present, Sam Redwine, a software engineer, at that time Assistant to the President and CEO of the Software Productivity Consortium, Herndon, VA. He is now an Adjunct Professor, Information and Software Engineering, George Mason University. Highly respected within IEEE-CS, Redwine may have been present simply because he was in town and knew a lot about IEEE procedures.

[43] While the minutes give the impression that these three axes were the work of SEEPP, Gotterbarn claims that Barry Boehm suggested them during discussion. (Emailed comments on draft of chapter, Gotterbarn, March 24, 2003.) Gotterbarn's claim may provide a good example of the unreliability of memory. Gotterbarn's Archive contains an undated document (SEEP1994-96, STEER.M1A), apparently the written report Gotterbarn and Melford submitted to the Steering Committee in advance of the May meeting (the computer file bears the date of May 27, 1994, one day *before* the meeting). This document explicitly discusses the "three axes". So, if the file date is right, then the axes are (one of) Gotterbarn's contributions, not Boehm's.

[44] Gotterbarn email to Davis (March 24, 2003).

[45] As Gotterbarn (March 24, 2003) recalls events, Zweben was to help draft the release because he had raised several objections to the version Barbacci had presented at the Steering Committee meeting. The minutes give no indication of this. Gotterbarn also recalls Barbacci saying that he was still seeking funding to help support SEEPP's work. The minutes make no mention of that either.

[46] Gotterbarn, who already had an airline ticket to Boston to attend a conference, only billed ACM for $60 to break his journey in DC and $22 for cab fare to and from Washington National. Gotterbarn Archive\SEEP1994-96\ACM expenses.

[47] Gotterbarn (March 24, 2003). Gotterbarn thinks Melford knew more about the event than he did. Gotterbarn (June 5, 2003, comments on Ch. 2).

[48] McFarland has B.A. in Physics from Cornell University (1969); M.A. and Ph.D. in Electrical Engineering from CMU (1975). He joined the Jesuits in 1975, and was ordained in 1984. He worked as a consultant at AT&T's Bell Labs in Murray Hill, New Jersey, for two years (before going to Boston College). He is now President of the College of the Holy Cross, Worcester, Massachusetts. His best-known paper in ethics then was: "The Public Health, Safety, and Welfare: An Analysis of the Social Responsibility of Engineers", *IEEE Technology and Society Magazine* 5 (December 1986): 18-26, already reprinted in Deborah G. Johnson, *Ethical Issues in Engineering* (Prentice-Hall: Englewood Cliffs, New Jersey, 1991), pp, 159-174.

[49] This assignment does not appear in the official minutes Melford prepared (Gotterbarn/94-95 misc/Boston 6-94), only in Gotterbarn's 1999 reconstruction of the preceding six years of work (addressed to "Leonard" Tripp). Melford's minutes merely conclude with the words "Preliminary assignment of working group chairs" (suggesting that matters may have been a bit less settled). Melford's minutes also omit Institutional Support altogether, defining Professional Relations to include it—"P-P (support for whistle blowers). P-org, P-individual, P-society, P-electronic community, obligation to foster responsible behavior in the non-professional". It is, however, easy to see how Institutional Support might have been carved out of this collection of topics.

[50] Interview of Melford, November 1, 2002.

[51] Michael C. McFarland, "Urgency of ethical standards intensifies in computer community", *Computer* (March 1990): 77-81; and Michael C. McFarland, "Ethics and the safety of computer systems", *Computer* (February 1991): 72-75. These two papers appear in the "Standards" section of the magazine, with Fletcher J. Buckley listed as "Editor"; one of Melford's papers is cited in the second of McFarland's. (McFarland was one of Melford's recruits.)

[52] Interview of Little, June 11, 2003.

[53] See his curriculum vita: http://www.uis.edu/~miller/keith.html (June 28, 2004)..

[54] Interview of Sullivan, November 10, 2002.

[55] Suzanne P. Weisband, Associate Professor of Management Information Systems, School of Management, University of Arizona. B.A. in Organizational Psychology (San Diego State University, 1979); Ph.D. in Social and Decision Sciences and Policy Analysis (Carnegie Mellon University, 1989). She is not a programmer but an expert on the "human factors" side of computing. http://www.bpa.arizona.edu/~mis/faculty/sweisband.shtml (August 28, 2004).

[56] S.P. Weisband and S.E. Goodman, "International Software Piracy", *IEEE Computer* (Nov. 1992): 87-90; and S.P. Weisband and S.E. Goodman, "Subduing Software Piracy", *Technology Review* (Sept./Oct. 1993): 30-33.

[57] Gotterbarn \by year\1993-4\Establishing the CFP (November 5 version). In an *October* 11, 1994 email, Jon Fineman (Gotterbarn Archive\SEEP1994-96\HELPOFR) says it was out in *IEEE Computer*: "Sorry to hear the world is giving you a hard time. I am confused though, what I was responding to looked like a call for participation that was advertised in the IEEE Computer magazine. That's what generated the note, since I did not want to miss the boat. Do you need any help getting things organized." Burnstein's (paper) file includes an even earlier version of the Call, a "Press Release from Melford" dated June 14, 1994. Gotterbarn was doing something similar. See Interview with Rogerson, February 24, 2003 (quoted below).

[58] See, for example, Interview with Rogerson, February 24, 2003: "On 19 October 1994, I received a broadcast email [of the Call for Participants] from Don Gotterbarn (sent through the Computer Ethics Institute list, to which I belonged). I'll give you a copy. I sent in a filled out form the next day." I never got the promised copy.

[59] The only evidence (in the archives) that Barbacci in fact did obtain a budget for SEEPP is a request from Gotterbarn in a letter (February 21, 1996) addressed to Curtis Harris at ACM requesting reimbursement for expenses of a meeting attendance at which had been authorized by Dennis Frailey. Gotterbarn\SEEPP1006-97\SEPBIL.

# Chapter 4: Failing—By the Book, 1995

Straight man: "How's your wife?"
Comedian: "Compared to what?"

## 4.1 The Volunteers

On January 1, 1995, SEEPP still had almost eleven months to complete its work. If it worked at the speed Anderson's committee had, it could easily hand the Joint Steering Committee the document promised by the date promised in the schedule drawn up the January before. SEEPP also seemed to have the resources to complete the work on time. The Call for Participation had brought in at least thirty volunteers.[1] Those, combined with the seven members of the task force, amounted to four times as many volunteers as Anderson had had (or, at least, officially recognized in the published code). And, as a group, the volunteers seemed as impressive in their credentials and as serious as Anderson's—and a good deal more varied.

Of the 29 who had filled out the application, only 17 were academics (and three of these were graduate students).[2] The remaining 12 were a mix of independent consultants, (high ranking) employees of large corporations, senior officers of smaller software companies, and a few miscellaneous enough to list individually. One was a full-time employee of a government laboratory completing a graduate degree at the same time. (One of the academics was also employed in a government laboratory while on leave for the year.) There was one philosopher (Fodor), one lawyer (from a major DC law firm), one business ethicist (Ernest Kallman, Bentley College, an ACM member), and even one "retiree". Fourteen volunteers were members of both IEEE-CS and ACM. Four were members of IEEE-CS only; two, of ACM only; and the remainder who answered the question (seven), of neither. (Two of the 29 ignored the question.)[3]

There was also considerable geographical diversity. The American addresses range from Louisiana to Minnesota, from Alaska to Florida, from Massachusetts to California. Tennessee, with two (*not* counting Gotterbarn), seems overrepresented. (The only other locales with two volunteers are: California, Texas, and Washington, DC—counting its Virginia suburbs.) Three of the 29 volunteers were British (English, Scot, or Welsh), two Irish, one Egyptian, and one Canadian (that is, nationals of the country *and* resident there). The diversity runs a bit deeper than the addresses suggest. So, for example, one of the British volunteers (Narayana Jayaram), though long resident in Scotland and England, begins his description of himself as "hailing from India" (where he had been born, raised, and received his first degree in engineering). Perhaps the British would count him as an Indian (though he was then Chair of the IEEE-CS Chapter in the United Kingdom). There is similar hidden diversity in those counted as Americans. One, for example, describes himself this way (and so might offer more of an international perspective than the simple description "American" suggests): "I grew up in a developing country (the Philippines) and set up a computer science program under the mathematics department in my alma mater (Ateneo de Manila University) back in 1981. I have gone to school both in France (University of Paris VI) and here in the US (Johns Hopkins University) so I have a unique perspective."

Most give their reasons for volunteering, along with their qualifications, in response to the application's question about "interests". Their reasons for volunteering vary a good deal but generally combine experience of software engineering with a commitment to certain standards. So, for example, one of the graduate students (Florida Atlantic University) explains:

> I am currently a Ph.D. candidate in Computer Science specializing in software measurement and software quality. Reliability and Safety are important quality factors. I also have personally studied the Challenger disaster in depth from the perspective of ethical issues. I have about 20 years of software engineering experience in industry (military systems and health care data processing).

Some of the applicants echo specific worries about competence described in Chapter 2. So, for example, one employee of a large company (in Missouri) explains:

> My interest is in the furtherance of SW Engineering as a professional activity. My observation is that ANYONE and EVERYONE seems to believe they can develop SW applications. Many of these follow no consistent process or provide any supporting products to aid in maintaining these applications. This gives legitimate SW professionals a bad name. I believe that ethical standards may provide an approach to solving this problem. I have 20 years' experience in developing real-time applications for visual simulation applications. I have dealt with a number of these semi-SW professionals and believe that this experience will be useful to the ethical standards effort.

For those applying from outside the United States, the reasons offered seem much the same. Here, for example, is the explanation offered by the one Egyptian (Amr El-Kadi, about whom we shall hear more later):

> I deal with the education of future SEs and I believe that it is of great importance to teach them RIGHT early on. I used to teach Doctoral level courses back in the states and many of my students were part timers working for big organizations. Many of them did not believe in SE and were still relying on "ad-hoc" techniques not only in civil applications, but in mission critical applications as well! I also do private consultation for industry. So, I see both ends: academic and practice. Enforcing acceptable ethical/professional standards on SEs I feel is part of my duty as a scholar and a practicing SE.

As a group, the volunteers seem to have a strong commitment to SEEPP's work.

4.2 "Documenting"

All this suggested that SEEPP would soon complete its work as successfully as Anderson's committee had. Against this hopeful reading must be set at least two negative signs. The first was that the working groups, though now organizing, still lacked the "Operation Guide" that the IEEE required for developing technical standards. That is, they lacked guidance concerning how to proceed. Since the Call's 3.3 declared, "The conduct of each working group

shall follow the same written procedures as govern internationally recognized IEEE Standards Activities", the working groups could not (officially) begin work until they had the Guide to follow. Melford was working on the Guide. He would have to complete it much faster than he had the Call—or SEEPP would certainly miss the scheduled November date for delivery of its "product".

The second negative sign concerned what was to be delivered. The November 1994 meeting of the Joint Steering Committee revealed some confusion (or, at least, an unresolved difference of opinion) about that. Some on the committee wanted a "standard practices document", that is, a document that would consist of relatively specific directives: "When you are faced with this particular situation, this is the technical approach you should take." (Standards documents can be dozens, even hundreds, of pages long.) Others wanted a short general document (a conventional code of ethics, a few pages at most).[4] This confusion was not new. It was already present on January 29, 1994 when SEEPP was named. Adding the "PP" for "and Professional Practice" had suggested something more than a code of ethics without defining what that "something more" might be (and, indeed, perhaps suggesting no more than that the code was to guide *professional* practice). The same confusion about what to deliver seems present in the Call. The general Scope statement 1.0 spoke only of "accepted principles for identifying and resolving ethical conflicts" (presumably, a conventional code of ethics) but the Global Application section 1.1 concluded with what sounds much more specific—"these standards shall also document various recommended practices and guidelines when no clear consensus can be established." "Practices and guidelines" certainly sounds like "this is the technical approach you should take".

Related to this confusion about specificity is another: what does it mean to "document" a practice? On one interpretation, documenting might be a descriptive undertaking, something requiring large surveys or other detailed investigation of what people actually do, say, and think as they work on software (the sort of survey that the body of knowledge task force was just then undertaking). A few of the applicants seem to have understood "document" in this way, mentioning their own empirical research in their description of "interests". So, for example, Kallman points out that he is "currently performing empirical research [on privacy] with Jeff Smith at Georgetown". Another academic (from Utah) does much the same: "I have been conducting research into ethics attitudes and practices among information systems professionals for about five years."

Most applicants, however, say nothing to suggest that they need look any further than their own experience to "document" (or at least set down) appropriate standards. So, for example, an AT&T vice-president (in New Jersey) simply states his "interests" this way:

> I have managed software projects for twenty-five years and written and spoken on software ethics and project management as well as network management. I am interested in how we may certify software managers, global development software, software safety and software stability.

Most of the academics presented themselves in much the same way, for example:

> As Chair of the Electrical and Computer Engineering Department at the University of Texas at Austin, I have a considerable interest in this topic. Further, I have implemented an undergraduate and graduate program in SE over the past two years and we are presently implementing an executive M.S. program in SE, for industry.

Another Texan (a grad student) also offers his interest and experience as his qualifications for the work of "documenting":

> I have been involved in computer security since 1986 when I was assigned to the US Air Force's Computer Security Office in San Antonio, TX. I served there as the chief of the R&D Branch and as chief of the Network Security Branch. I was then assigned to the USAF Academy where I taught in the Computer Science Department. I am currently a Ph.D. candidate at Texas A&M University working in the area of computer security. I have been the author of several papers on computer security as well as a recent paper on ethics. I am currently working on a textbook on computer security with a tentative completion date of 15 Jan 95.

4.3 Another Plan

About the time this Texan was supposed to deliver his textbook to the publisher, the first of the two negative signs disappeared. Melford delivered a draft of the operating guide to Gotterbarn. Gotterbarn sent back a few suggestions for improvement and then emailed the guide to his own working group (Professional Competence) on February 22.[5] The email takes the form of a letter beginning with the date followed by "Dear SEEPP Volunteer". After thanking the volunteers for their interest, Gotterbarn admits that "[the] work of the task force has been slow to start due to some of our unfamiliarity with this process". After apologizing "for the delay", Gotterbarn takes a paragraph to describe how the working group fits into SEEPP's work and that of the Joint Steering Committee's two other task forces. Part of the description justifies the division of the task force into working groups as an instance of "good computer problem solving technique" (divide the task into several subtasks that can be distributed so that no one has a task too large to handle). Gotterbarn then identifies a "problem that has caused some of the delay", that is, resolving "the question of how to mange and organize the working group process", and (at last) announces, "I have a plan."

The plan has two parts. The first, briefly described, is simply a "small email list" (ProfComp@ETSU-east_tenn_st.edu, a "manual listserv"). The list was to allow each member of the Professional Competence working group to communicate with the rest of the group without revealing the writer's individual email address. Located on a server at East Tennessee State, the list would not resend automatically (as a true listserv would). Gotterbarn would "moderate" it, that is, pass on any messages properly directed to the list after screening out "spam" (junk mail) and "other irrelevant activities". Any mailings to the list address would go only to the list. Gotterbarn also wanted members of the working group willing to yield a little of their privacy to be able to communicate with each other without going through him. He therefore promised to circulate a list of all current members of the working group. Anyone who did not want to be on that second list could notify him "within two weeks". He would send out the preliminary list on

March 5, asking for corrections. He also promised that this second list "will not be circulated to any other organization nor will it be made available by me to any other group."[6]

The second part of Gotterbarn's plan was more substantive, a "general outline for the development of our standards of professional competence". This outline, "written in Jello", had three steps: 1) gather information, 2) analyze it, and 3) synthesize it in a "statement of scope". The information would take the form of digests of "other profession's standards of *practice*" (my italics). Each member of the working group was to send to the "LIST" the name and address of any organization known to have standards of professional practice, as well as an indication of willingness to get copies of the standards (preferably in electronic form) for the working group and willingness to write a "short abstract" of that standard "indicating the scope of application of the standard, means they use to update it, how practitioners are educated about that standard, and the methods of standards enforcement, if any." Given this information, Gotterbarn suggests, "[we] should be able to abstract a model of professional practices for our use." This "model" is, apparently, the "synthesis". It must then be put into in a "Scope" (a statement of "what it is we think we are doing") and a "structure" in which the working group would "organize our suggestions and conclusions." Once the model is ready, it should be possible to "subdivide the problem" and ask for volunteers to write rough drafts of the sections of the document (with all drafts circulated to all members of the working group, an "electronic meeting").

There are three features of the plan worth noting now. The first is that Gotterbarn has switched from talk of "professional *competence"* to "professional *practice*". Gotterbarn has, it seems, invited his working group to do what SEEPP (as a whole) was supposed to do. Unable to "divided and conquer", he was now trying a more holistic approach. Second, the holistic approach was to be temporary. Once the working group had a "model" to work with, it (not SEEPP) would divide into several drafting committees (one for each section of the code). Third, Gotterbarn continues to be a "hands-off" manager. He is still trying to delegate the work of drafting to others.

Following this "plan" are ten single-spaced pages, the long-promised "Guide to Operations". Divided into a page-long table of contents and five numbered sections, beginning with "Overview" and ending with "Style", the Guide is a document of imposing detail. The details provide evidence that Melford had learned from the advice the experts had given him. For example, the Overview begins by observing that the "IEEE-CS/ACM standards for software engineering ethical and professional practices…are *not* being formulated under the auspices of a formal IEEE Standards Project Authorization Request (commonly known as a 'PAR')." The standards need only be "developed in accordance with *appropriate* IEEE and Computer Society Standard's Policies and Procedures." (Italics mine.) The IEEE Standards process was not to be a straightjacket. SEEPP could have the advantages of the IEEE Standards process without the disadvantages.[7]

The IEEE process nonetheless weighs heavily on the document. Six of the ten items in "References" (section 2) are IEEE publications concerned with IEEE Standards. (The remaining four are generic: *Robert's Rules of Order*, *The Chicago Manual of Style*, a *Handbook on Nonsexist Writing*, and *Webster's New Collegiate Dictionary*.) Section 3 (immediately following the references) consists of "definitions". Subsections 3.1-3.3 are concerned with defining "IEEE standards", distinguishing between "standards" (with a small "s"), "Standards" (with a capital "S"), "recommended practices" (marked by a "should" rather than the "shall" of a Standard), and

"guides" (more than one practice can be recommended). They also relate these distinctions to ACM practice. Section 3 concludes with a list of fourteen acronyms (including "WG" for "working group"). Section 4, by far the longest (about six and a half pages), sets the procedures for working groups (meetings, quorum, voting, the first draft, Scope, outline, appropriate title, membership requirements, expiration or revocation of membership, elections of chairs and vice-chairs, balloting, and resolution of comments, objections, and negative votes). The Guide's procedures are, more or less, the IEEE procedures for writing standards. They are quite detailed. Consider, for example, "4.1.4 The first draft":

> After establishing a draft Scope (see clause 4.1.4.1), the WG should outline the important issues and sub-topics it thinks the standard should address (see clause 4.1.4.2). The WG should also establish the nature of the standards it is to develop (i.e., standard, recommended practice, guide, etc., see 4.1.4.3). Although it does not have to be included in the text of the standard, the WG should clearly and concisely define the purpose of the document. This generally describes "why" the document is being developed. For example:
>
> > "Purpose: Full-range current limiting fuses are not currently covered by standards. In fulfilling this need, the working group will also define and recommend an overall action plan to clarify and improve industry understanding for all types of fuses."
>
> The WG Chairperson should then identify an individual to author an initial draft. The author should be permitted to prepare the draft with or without additional input or assistance from any other WG members, at his or her discretion. The consensus of the WG membership should then be evaluated to determine if the complete draft can subsequently serve as the foundation for an iterative process of refinement.[8]

4.4 Too many meetings

Since this memo went out on February 22 (with that date), Gotterbarn and Melford had substantial progress to report to the Steering Committee when. five days later, it met in Nashville (where the ACM was also meeting). The working groups had been set up. The Guide was (more or less) complete and distributed. At least some of the working groups had begun work. There was still plenty of time to complete a draft of the code of ethics before the November deadline. No surprise, then, that the emphasis of the Steering Committee's half-day meeting was not SEEPP but (again) the Body of Knowledge Task Force (now officially so called). What is surprising is that, after the three task forces had reported, the Steering Committee moved into an hour-long executive session—an executive session that excluded the ex officios. Though we lack minutes, we know something of what happened.

Late in 1994, Pat Douglas, chair of the Body of Knowledge Task Force, had presented the Joint Steering Committee with a rough draft of her task force's initial report, the plan for a survey of practitioners to see what knowledge they thought necessary to be a competent software engineer. The Steering Committee had to review the plan before the survey could begin. Some

members of the Committee found the plan potentially controversial enough that the chair scheduled the executive session after the plenary meeting. Douglas thought the executive session, though unusual, was understandable. Her task force was to decide what should be taught to would-be software engineers and (in effect) who should teach it. For academics, that meant student enrollments (jobs) for some and loss of enrollments (and perhaps loss of jobs) for others. What Douglas did not find so easy to understand is what happened during the executive session. From questions asked of her later, she concluded that, in her absence, the Committee had assumed that the survey would be 60 pages long, a survey so detailed that nobody was likely to fill it out. Somehow her presentation the hour before, and the questions that followed, had not prevented the Committee from confusing her written report (seventy pages long) with the survey it discussed. Secrecy has a price, a price SEEPP never had to pay.[9]

On March 25, 1995, Gotterbarn, Melford, Barber, and Miller met in Washington, DC. They completed the Operations Guide and also started on the task force's "Scope" (which they hoped would serve as a model for the Scope each working group would also have to write). Gotterbarn and Melford also agreed to meet again, in April, during the meeting of the SEI's Faculty Development Workshop in New Orleans. At that meeting, they did no SEEPP work but agreed to meet again at the National Education Computing Conference (NECC) in June.

But once home, Gotterbarn changed his mind. Of the twenty-two months SEEPP once had to write the code of ethics, it now had only six, much less time than Anderson's committee had needed to write the ACM code. At this rate, SEEPP could not meet its November deadline. Early in June, Gotterbarn wrote Melford to explain what he had decided.[10] The letter began (after the simple salutation "Bob") by announcing that Gotterbarn had cancelled the June meeting because there would have been only four in attendance (Little and Barber in addition to himself and Melford). Gotterbarn then turned to what was really bothering him ("a major problem"): "We are just plodding along and have not given adequate direction, advocacy, or leadership by example to the group." What Gotterbarn meant, he said, was that "We schedule meetings but the preparation is not complete…and we leave with tasks to work on between meetings [that] are not completed by the next meeting. With the exception of the draft CFP and the draft guide to operations, work seems to be done ONLY at the meetings." To support this claim, Gotterbarn summarized the major events in SEEPP's history, taking almost a single-spaced page, beginning with the meeting of April 30, 1994. He then returned to his main point:

> We have meetings and articulate great plans but very little happens. We have many volunteers from September-November of 1994 who are complaining [—] wondering when, if, we are going to get started. Some of those who originally volunteered for working groups have already abandoned ship. I am afraid the same is happening with the task force itself.

Gotterbarn concludes by asking Melford to email "the material we worked up in Washington", promising to write a cover letter to the volunteers "with some directions on how to get started, a plea to contact their groups, etc."[11]

A few days after he wrote that letter, Gotterbarn emailed a list of the working group's members, including mail and e-mail addresses, asking recipients to "check for accuracy" (June 10, 1995). There were twenty-four names on the list. Two were new. Eight of the originals were

gone. Among the first corrections to come in were three from the Illinois Institute of Technology (IIT). Behind these corrections lay a sequence of events extending back six months, events that offer a foot-soldier's perspective on SEEPP's work.

## 4.5 An idea for research

On a cold but bright Chicago morning early in December 1994, Vivian Weil, Director of IIT's Center for the Study of Ethics in the Professions (CSEP), sat at her desk reading a photocopy of the three-page Call for Participation printed in *Computer and Society*. Someone in IIT's Department of Computer Science had sent it to her because CSEP, one of the oldest ethics centers in the world, was one of the few with a focus on engineering. It was also one of the few with an enduring interest in codes of ethics. CSEP has two external functions: 1) to do research concerning professional ethics; and 2) to aid the professions by making its knowledge of professional ethics, its research, and its teaching available to the professions. From its beginning in 1976, CSEP had collected codes of professional ethics more or less systematically. By 1994, it had several hundred on file. Those files might be useful to software engineers trying to write a new code of ethics. (Recall that one of SEEPP's first plans, never carried out, was for a depository of relevant codes.)

Then CSEP's Senior Research Associate, I had written a half dozen or so articles on codes of ethics.[12] I thought a code of ethics was central to organizing an occupation as a profession. Weil, on the other hand, was less certain. She tended to think of codes as a means of education in already existing obligations. That morning, though, she was wondering what those who write codes think they are doing. It might be interesting to find out. Here was the opportunity, an opportunity for CSEP both to do interesting empirical research and to make itself useful to a new technical profession. Weil did not know much about software engineering, indeed, until reading the Call, she had, she believes, never heard the term. But, after almost twenty years doing professional ethics, she no longer worried about starting out ignorant of a discipline. What was important was having the resources to learn what one needed to know. For software engineering, the resources to learn seemed readily available. IIT's Computer Science Department taught several courses in software engineering. A phone call had given her the names of the two faculty who taught the courses. So, Weil decided to put her thoughts on paper. The result was a concept statement of about 500 words, "a Project on the Creating of Software, Professionalism, and Responsibility."[13]

Over the next few days, Weil phoned people she knew who might know something about software engineering generally or the joint IEEE-CS/ACM project and repeatedly rewrote the concept statement to take into account what she had learned. By the second week of December, she felt ready to bring together the people she had identified as interested, locally available, and potentially helpful. On December 15, they met: Ilene Burnstein (Computer Science, IIT, an expert in software development process); Barry Weiss (a Chicago lawyer interested in liability for software failure); Ullica Segerstrale (Social Sciences, IIT, with an interest in science studies); and me (expert on codes).[14] What became clear at the meeting was that there was interest in doing something related to the concept statement but no agreement on what in particular. Burnstein seemed to be interested in technical standards; the lawyer, in issues of responsibility; and I, in writing a code of ethics; but we eventually agreed to focus on Miller's working group

on Reliability and Safety. Weil revised the concept statement accordingly. There were several more meetings, with Bogdan Korel (a colleague of Burnstein's with an interest in software testing and debugging) making the research group six. The three with enough time (Burnstein, Weil, and I) became "the research group"; the other three (Korel, Segerstrale, and Weiss) became "consultants".

By the middle of January, Weil thought the concept had developed far enough that she asked me to contact Gotterbarn, with whom I had shared a stage once or twice, to see whether SEEPP might be able to use CSEP's resources and whether CSEP could study SEEPP's work. Gotterbarn responded enthusiastically, even putting me on the Professional Competence list (as a courtesy because he thought my interests closest to his own). I therefore received the long email of February 22 that delivered the "plan" and Guide. The day after that email, CSEP submitted a proposal to the National Science Foundation (NSF) for a fast-decision Small Grant for Exploratory Research. The proposal was fast-tracked because SEEPP's schedule did not allow for NSF's normal six-month review. SEEPP was supposed to complete its work before a normal review could have determined that IIT's would-be participant-observers deserved support. (By February, CSEP had also received Miller's enthusiastic invitation to participate in the Reliability and Safety working group.)

Since the proposal was funded two months later, it is worth noting that it contained some significant errors. The first is in the Project Summary. It explicitly uses "SEEPP" as the acronym for the Joint Steering Committee. This suggests that IIT's research group had a poor grasp of the distinction between Joint Committee, task forces, and working groups. I certainly did. The body of the proposal contains other evidence of that. So, for example, the fourth item under "outcomes" is:

> Because the Joint Committee has adopted an open process, allowing everyone to read the committee debate (on e-mail) and to respond, we hope to participate in that debate as well as monitor it, not only asking questions both substantive and procedural (for our benefit) but asking or answering questions for the benefit of the important work the Joint Committee is engaged in.

There would be little to object to in this sentence if it had referred to "SEEPP" rather than to the "Joint Committee". SEEPP had indeed adopted a (relatively) open process. Anyone in the world with an interest could join one of SEEPP's working groups and the deliberations of a working group were open to all its members. That was, however, not true of the Joint Committee. Neither its membership nor its emails were open to the world; indeed, some of its deliberations were closed even to its ex officio members. How many other volunteers were as confused about SEEPP's organization as IIT's research group?

This confusion may explain in part why the IIT volunteers sought to "correct" Gotterbarn's June mailing list for members of his working group (Professional Competence). Gotterbarn had not made a mistake. IIT's volunteers (Burnstein, Weil, and I) did not belong on *that* working group's mailing list (as members). We belonged on the mailing list of Miller's working group (Reliability and Safety). Even though I was on the Professional Competence email list, I did not (technically) belong on the list of members. I had been added to the list as an

observer. I too had joined only Miller's working group.[15] Four months of silence (February to June) had been enough to wipe out all memory of Miller's working group.

4.6 Out of nowhere

Confusion about SEEPP's structure, while explaining part of the reason for "correcting" Gotterbarn's June list, is probably not the whole explanation. Underlying the confusion was a growing unease (akin to what provoked Gotterbarn's "Bob" letter). IIT's research group had officially begun monitoring Miller's working group's email in April, as soon as it was clear that NSF would fund the research. There had been nothing to monitor. The first SEEPP email to arrive in over two months was Gotterbarn's on June 10. Our research group assumed we had to be on the June 10 list of volunteers to receive the emails we were supposed to monitor and that we would not have received this email unless we were members of the working group in question. We had (I think) confused SEEPP with one of its working groups. We did not realize that we (or, rather, I) had received the members list because, and only because, Gotterbarn had put me on the Professional Practices list as a courtesy (even though he had not put me on its members list).

Encouraging this misunderstanding was another email that came the same day (June 10, 1995)—and, indeed, was printed as if it were continuous with the first. Again, it was another form letter and again I was the only one in the IIT research group to receive it. While the emailed list had been titled "Volunteers to the professional competence working group", this email addressed its readers as "Dear SEEPP Volunteer".[16] It promised action:

> Sometime back I sent a kick-off letter for our effort. The letter, appended below [email of February 22, but dated February 16 and without the Guide], gave a complete approach for the development of our standard. It was probably not a good idea to list all of the tasks at once. No one knew where to start. To start we need to have some folks jump in and pick up particular tasks. I have listed them below.

There were three "tasks" on the list. The first was to develop a "straw man", that is, a model of a "professional competence standard" that can be presented to the working group for comment. "Do we have a volunteer or two?" The second task was to "examine and summarize other professional standards." Attached to the end of the document was a list of examples of these "standards" (with contact information): "BRITISH COMPUTER SOCIETY CODE OF PRACTICE, 1 SANFORD STREET SWINDOM, WILTSHIRE SN 1HJ; PROFESSIONAL STANDARDS—ACCOUNTING AMERICAN INSTITUTE OF CERTIFIED PUBLIC ACCOUNTANTS, INC. COMMERCE CLEARING HOUSE, INC 1997….[and so on]" Most were codes of ethics of professional societies (whether titled "code of practice" or "code of ethics"). The third of the tasks Gotterbarn set was "be original". Anyone with a specific idea about what should be a professional practice should "write it up and send it in to the list". The letter concludes, "This is the first step in item A below ["Gathering information" in the February 22 email]. I look forward to hearing from you. Don Gotterbarn".[17] (Since the only SEEPP volunteers likely to have this list of codes would be those in his own working group, addressing

this email to “Dear SEEPP Volunteer” may have added further to confusion of task force and working group.)

The hoped-for action was slow in coming. The first evidence of it arrived at IIT on August 31—from Egypt. Eight pages long (single spaced), Amr El-Kadi’s email summarized two codes of ethics and also reviewed Microsoft Certification for engineers. This email was not only the first evidence of the hoped-for action to arrive at IIT but also the last until early November—when suddenly one email after another came in. By then, of course, it must have been obvious to everyone that the task force would miss the November deadline. But the emails that began to arrive in November suggested something more serious. The first (November 3, 1995) bore the subject heading “Are We Still Working?” It was directed only to the Professional Competence working group:

> Volunteers,
>
> I haven’t received any messages since July. I have sent a couple without response. Are we still working??
>
> I hope I got your names correct.
> Ed

For the research group at IIT, this email was full of mysteries. The first was that only Weil and I were on the list of recipients. For some reason, Burnstein, the third member of the research group, was not. The second mystery was that “Ed” had sent “a couple” of messages “since July” without response. No one in IIT’s research group had received any of them. Indeed, with the exception of El-Kadi’s code analysis, Ed’s email was the first that IIT’s research group had received from SEEPP (or anyone involved with it) since June.[18] What had IIT’s research group missed in July? What had it missed in August, September, and October? Had “Ed” used the same email list he was using now (and had IIT’s computer or the ether eaten the mail) or had he changed his email strategy? Third was the identity of this “Ed”. His name had not appeared on the list Gotterbarn circulated in June. That was odd because the mailing list “Ed” was using (23 names along with the profcomp listserv and the task force’s se_ethics listserv at the Software Engineering Institute) was otherwise familiar. Why should we know everyone but “Ed”?

After a meeting during which the research group meditated on all this, Weil emailed “Ed” (November 14, 1995):

> We have a small NSF grant to study your process of developing/formulating standards. We have seen no messages since July. Please let us know if we have missed any. Also please let us see any replies to your message of November 3.
>
> Vivian Weil and Michael Davis

A response came the next day “From Ed Mechler, CCP” (November 15, 1995). Mechler was plainly someone not to waste words—or to fail to say what needed to be said:

Vivian and Michael,

Are you members of this Working Group in SEEPP?
Is NSF the National Science Foundation?
Why weren't we informed of your observations?

So far out of 26 listed members with a possible 2 or 3 cc to other groups, I have received 3, including yours, that don't belong to the working group and 3 from members wondering the same thing I am. I will send you copies when I find out more about you, ethics you know.

How is the weather in Chicago? I have a son who just graduated Penn State as an Arch Engr and is working for Turner in Chicago.

Talk to you later.

Ed

Mechler's first question came as a surprise. IIT's research group thought it did "belong"—and it had received Mechler's email because it did belong. Of course, as explained earlier, the research group thought it belonged because it thought it had joined Gotterbarn's working group (because it had joined SEEPP, because it had received emails sent to Gotterbarn's working group, because Gotterbarn-co-chair-of-the-task-force and Gotterbarn-chair-of-the-working-group were hard to keep straight, and because Miller's working group—which it had originally signed up for—had been so inactive that it had disappeared from memory).

The second question (about NSF) told IIT's research group that Mechler was probably not an academic. The third, however, added to a worry that Mechler's first email had engendered. SEEPP now seemed much less organized than Gotterbarn's emails had made it seem. IIT's research plan depended on SEEPP's organization, especially on most communication going through ETSU's profcomp list and being available to everyone (including the IIT participant-observers). It also depended on everyone in our SEEPP working group knowing we were monitoring its emails. Clearly, Mechler was now receiving emails about the working group that were not going through the ETSU list. That was why Weil had asked him to share with our research group the emails he received in response to "Are We Still Working?" Had he shared the emails at the time (instead of several years later), the research group would have been even more worried. The emails that Mechler received between November 3 and November 15 suggested considerable disorder in SEEPP communications.

The first, sent almost as soon as Mechler's arrived, came from "Manny Norman, Sr. Sys. Progmr, LT" (staff, Eastern Michigan University): "I'm still here, but you are the only person I ever hear from."[19] El-Kadi responded the next day: "I really don't know what to say except that I share your worries: Are we still working, or did someone simply decide to terminate the task force without informing the people who were willing to work?!"[20] Pat Sullivan responded: "didn't get previous message—will go back through what's come from SEEPP (I haven't

received anything since last summer, either). Otherwise, as far as I know, yes, it seems but slowly."[21]

The other two responses were even more worrisome. One, from an assistant professor (and Major) at the Air Force Institute of Technology at Wright Paterson, denied membership in the working group: "I was helping Pat Douglas on the Body of Knowledge work. Did that get me on your list?"[22] All Mechler could reply was "your name and address has been on the mailing list for the Software Engineering Ethics and Professional Practices, the Professional Competence working group since June. We just have not heard anything since July so I sent the message below to inquire."[23] Mechler sent the same response to Ernest Kallman (November 14) after he wondered why Mechler had written him (November 11). Now able to appreciate the point of Mechler's question, Kallman wrote back immediately (November 14): "I understood that I was in the privacy group under the direction of Patrick Sullivan and have communicated this to Don Gotterbarn. I would prefer the privacy group to this area as I feel I have more expertise in it. If you communicate with Don, feel free to repeat this."[24]

Mechler now had six volunteers out of six responses.[25] But the IIT response had brought in three volunteers for two addresses (addresses not supposed to be for the Professional Competence working group at all). Two out of the six responses had denied having anything to do with Professional Competence (even though their names were on the members list). That was troubling enough. How many others listed as members had never been members? Were there others, like the IIT research group, who (it seemed) should have been on the printed list of members but weren't? This question about the printed list would not have been important if the ETSU list worked properly. But it did not seem to. Mechler had sent out his original message through the list as well as using individual addresses culled from Gotterbarn's lists and addresses from headers on emails. All responses came from individual addresses.

By November 9, Mechler had concluded that the ETSU list was not working—and may not have been working since July. He emailed Gotterbarn directly: "I have sent two e-mails plus ask other volunteers the state of the group since I have not received any mail back from the group e_mail address. I thought I would try to contact you from your original address since no activity since July. Please let me know."[26] Gotterbarn responded on November 14 (without acknowledging Mechler's concern that the ETSU list was not working): "There has been no activity because I have not been pushing it. I have some concerns with things that are going on and I want to be sure your efforts are going to bear fruit before I start to push. Thanks for hanging in there. I will get back to everybody as soon as I can."[27] Two days later, Mechler responded: "What are the problems with our project? Can I help?"[28] Mechler also quoted Weil's request to be allowed to see "replies" and asked, "Can I honor the request?" The archives contain no answer to this email. (Chapter 13.4 will deal with the ethical issue Mechler raised—as well as several others that developed later.)

## 4.8 Gotterbarn's Prison

On November 27, 1995, the official deadline for completion of SEEPP's work, El-Kadi emailed again, apparently copying his list from Mechler's or from Gotterbarn's, neither of which had Mechler's address or Gotterbarn's. Though the email is largely addressed to Gotterbarn, his name appears only on the "cc". Discontent till now smoldering had burst into flame:

> I think that Don does owe us all a response this time. As chair of the SEEPP Committee, you have contacted us and we have "volunteered" to work on this committee because of our strong belief in the cause.
>
> Since the time this activity started, everything has been so slow?! We don't even have a mailing list that works?
>
> Is there such a committee (SEEPP) or has it been terminated? If it still exists, what is going on? If it has terminated, then who did take that decision, why, and how come he/she never notified the volunteers as any professional would do?????
>
> Don, we are all waiting for your response.

Gotterbarn apparently appreciated El-Kadi's complaint, but took two weeks to respond. The two week delay seems to be explained at least in part by a December 11 email to Melford, SEEPP's co-chair, that begins much as the El-Kadi's had:

> I have had trouble contacting you. I hope this email gets through. I am sending it to your computer.org address and home address. Below are two messages I will send out by the end of this week so people can start to get some work done. And make plans for attending a meeting of the task force. Let me know what you think.[29]

Apparently, communications between SEEPP's co-chairs had also become a problem (quite independent of any list). Gotterbarn was now ready to solve the problem of Melford's veto by treating silence as consent ("will send out") rather than as "I forbid".

The first of the documents Gotterbarn put "below", a letter beginning "Dear SEEPPers", seems never to have been sent. It consists of six numbered paragraphs describing the steps by which the working groups should "get started" (a more detailed version of instructions sent out February 22).[30] The six are preceded by two introductory paragraphs worth quoting in full:

> I am concerned by our lack of progress. I am not aware of any working groups starting work. I know you have all been quite busy but we need to get started.
>
> In March several of us met in Washington and wrote a statement defining the scope of the task force and reworked the formal procedures of deliverables from the task force. Bob has these documents and will forward them to the group.

The message implicit in these two paragraphs is bleak indeed. None of the working groups is working—or ever had. (For some reason, Gotterbarn chose not to mention El-Kadi's August email or other work that his own working group had performed since July.) The deadline is passed for delivery and all that Gotterbarn can think to do is to plead that "we need to get started". How do "we" get started? Gotterbarn's only answer, the second paragraph (and the six steps to follow), is to continue to go by the (IEEE's) book. He will distribute the documents

drafted on March 25, 1995, and (apparently) finished a few months later. How important are these documents? Not important enough for Gotterbarn to have a copy of his own to work from. Melford will have to provide them.

Gotterbarn did not send this email to the list because Melford never responded to the request for approval. Gotterbarn still held back from actually treating Melford's silence as consent.

The other document "below" is a draft letter to the Professional Competence working group. Gotterbarn begins that one by acknowledging El-Kadi's letter "expressing concern about the lack of progress…of our group". Gotterbarn then apologizes for the lack of progress "especially to those who have already done work for the group". These preliminaries take two short paragraphs. The much-longer third paragraph tries to explain what went wrong:

> I have been concerned about the activity of the SEEPP task force. The last task force activity took place in March, when several of us met to finalize the task force guidelines and set the direction and scope of the task force. This material has not yet been distributed to the SEEPP working group leaders. As one of the co-chairs of the SEEPP task force, I have tried to motivate it from the bottom up by developing a plan of action for our working group and sharing that plan with other members of the SEEPP task force. Although you all started the work of our working group, there is very little happening from the task force and I did not feel right asking any more from you until I see some positive indications that your work will ultimately bear fruit.

At the time, this explanation seemed unhelpful for at least two reasons. The first concerned the factual premise. Gotterbarn said that the task force last met in March, completed the guidelines, directions, and task force scope, but "this material has not yet been distributed to working group leaders." Why, I wondered, weren't they distributed? Gotterbarn does not say. More important—at least for those of us with a good memory or an email file (whether paper or electronic)—was that what Gotterbarn said raised questions about the status of what *all* SEEPP volunteers had received by the email dated February 22, 1995 (and presumably had been trying to follow ever since). That email contained—beside yet another copy of the Call—what looked like "directions" (the second part) and "guidelines" (the third part). Was Gotterbarn referring to these documents or something else? What were these documents? What was their relationship to those completed at the March meeting?

Gotterbarn's explanation was also unhelpful because what happened to the guidelines, directions or scope *in* March, however confusing, could tell the working group nothing about why so little happened *after* March. "Some" members of the working group had (according to Gotterbarn himself) been able to do something. Why not others? Indeed, Gotterbarn's own June 10 email showed that, even after March, he could set "direction", specifying particular jobs to be done. The cause of the working group's current sad state certainly did not seem to be his method, "motivating from the bottom". The one fact he cited that did seem relevant, that there was so little going on in the other working groups after March (or, perhaps, after June) that he did not "feel right asking any more of you", would be relevant only once he explained why so little was going on in the other groups . He offered no explanation of that. I think ordinary decision theory

does offer an explanation (or, at least, a hypothesis worth investigating, especially for those thinking of learning from the mistakes of other code writers).

SEEPP's general strategy had been "divide and conquer". The strategy works well when one can "conquer" a domain piecemeal—when, that is, the whole consists of (more or less) independent tasks, each a worthwhile objective in itself. When, however, the tasks are mutually dependent, so that none is worth doing unless all are completed successfully, and the work is divided among several different people or groups, then "divided and conquer" creates (what decision theorists call) an "assurance problem" (a certain kind of coordination problem of which the "prisoners dilemma" is a close relative). The parts gain their value (in large part at least) from their being included in an organic whole. Half a code is not (much) better than none. If nothing anyone does will be worth much unless all succeed at their individual tasks, doing anything until all can (as Gotterbarn put it) "see some positive indications that [their] work will ultimately bear fruit" becomes irrational. Each must have assurance that others are doing their share before it is rational for him to do his.

There are at least three ways to provide that assurance. One is to make it rational for each person to do her part even if others do not do theirs. The easiest way to do that is to pay for the work piecemeal, an option not open to a task force of the IEEE-CS/ACM Joint Committee without external funding. So far, all SEEPP's attempts to get a grant from the National Science Foundation or some private philanthropy had failed. The second way to provide assurance is to take a chance, that is, to do a little work oneself, see if others reciprocate, and then act accordingly, quitting if others do not but taking another chance if they do reciprocate. Gotterbarn's working group had, in effect, been trying to solve the assurance problem this way since June ("motivating from below"). The other working groups had not reciprocated—perhaps because they did not know that anyone in Professional Competence had done anything. (In this respect, Gotterbarn's failure to boast to the other SEEPPers about what his working group had done was a tactical error.) This second way of solving the assurance problem requires good communication between groups—or the ability of each group to observe the progress of the others. Individual members of SEEPP had a very poor idea what other members of SEEPP were doing. The listserv was not designed to track work—and, even today, it is hard to imagine how it could be. The third way to solve the assurance problem is to reformulate the task so that coordination across groups is no longer necessary.

Gotterbarn soon showed that he had chosen a fourth option (a "technological fix"): "Please have patience, I am taking a more direct approach to the problem." He had set up a new mail list "for our group" (PRFCMP-L@UTKCM1.BITNET). He would now be sending emails through a server at the University of Tennessee, Knoxville, rather than through a server at ETSU. The server seemed to be working: "This is the way this message was circulated." The list would transmit messages automatically: "Messages sent to the list will be forwarded immediately to all members of the group….A reply sent to this message will be sent to everyone in the group." The working group would have a listserv rather than a mere list. (Apparently, Gotterbarn was no longer worried about spamming or other abuse of the list.)

With only the evidence Gotterbarn had before him on December 15, we cannot be sure what caused SEEPP's work to come to a near halt after March. What we can be sure of is that, given the cause Gotterbarn himself offered (the need for "positive signs that [their] work will bear fruit"), his "direct approach" had no more chance of success than his old approach—for a

least two reasons. The first is that in fact it *was* his old approach (except that the list was now automated). The problem existed even when the old list worked. The second reason his "direct approach" had little chance of success was that, even if it were new, it would not have solved the assurance problem. That problem could not be solved without visible progress in the other working groups—or their dissolution (or some conviction that completing a part of the work was worthwhile even if the whole work was not completed). Gotterbarn had no plan to improve the visibility of progress in other working groups, not even a common list for all of SEEPP, no plan to get the other working groups started except to try again to get his own group working, and no thought at all of dissolving the other working groups. Gotterbarn had, it seems, run out of ideas.

The paragraph announcing the automated list, when combined with his other emails for the year, suggests that Gotterbarn may have run out of steam as well. The paragraph concludes: "Three members of the group have already done some work in response to my July message. I will forward the material to you shortly."[31] Who were the three who had done something—or, rather, who beside El-Kadi had done anything? The material Gotterbarn promised to forward would have answered that question, but he seems to have sent no more emails to SEEPP or his working group in 1995. The next Gotterbarn email to arrive at IIT would be dated March 3, 1996 (and would not contain the promised material). The first SEEPP-related email to arrive at IIT after Gotterbarn issued his promise is dated December 28, 1995. While it was the sort of thing Gotterbarn had promised (extracts of the "AICPA Standards Manual"), it was not forwarded but seems to have come directly from its author, Manny Norman, passing through the new listserv—two weeks *after* Gotterbarn had counted to three.[32] Gotterbarn's never-sent "Dear SEEPPers" letter of December 11 contained a diagnosis of his own failings: "You have got to be ready to follow through. I have started my group TWICE but each time I did not follow through." Here, it seems, was a third failure to follow through.

Criticizing Gotterbarn's performance is not hard: He had a problem he did not know how to solve. He tried his best but his best clearly was not good enough. It is therefore worth pointing out that none of the other SEEPPers (that is, the chairs of the various working groups) had offered any alternative to Gotterbarn's plan though he invited alternatives by initially describing it as "written in Jello". Gotterbarn seems anything but a dictator. In addition, he had his own coordination problem. As one of SEEPP's two *co*-chairs, he had (he supposed) to work with the *other* co-chair. In the first year and a half of SEEPP, the two had worked together pretty well. Melford had done much. He had recruited about half the chairs of SEEPP's working groups, led SEEPP through the IEEE's serpentine procedures (preparing first drafts of the Call, Operations Guide, and Scope), and attended all but one of SEEPP's meetings (and shared with Gotterbarn in planning all of them). But, as 1995 wore on, Melford had become increasingly hard to reach. Perhaps Gotterbarn should not have been so quick to give up on the face-to-face meetings even though they seemed to accomplish little. One thing they had assured was that every month or two, Gotterbarn would get to consult with Melford face to face, disposing of whatever business had not been completed by email. Melford's own explanation of why he had become so hard to reach is that his consulting practice had becoming increasingly "crazy" (as the high tech boom of the mid-90s developed).[33] Whatever the reason, Gotterbarn was finding it increasingly hard to do anything as SEEPP's co-chair. He never sent his "Dear SEEPPers" letter to SEEPP because he believed that, according to IEEE procedures, he could not properly do so without the consent of his co-chair. On his own, he could only communicate with his own working group.

## 4.7 Nadir

December 21 is the darkest day of the year. We can easily imagine Gotterbarn looking out on the gray round-tops of east Tennessee on that day's late afternoon wondering whether he was presiding over a disaster. He was, after all, no closer to having a first draft of a code of ethics than he had been a year before. Indeed, given the dwindling of enthusiasm among the volunteers, he was, it seemed, further from a first draft. And there was nothing more he could do about it. He had tried everything he could think to do. Nothing worked. He was now repeating himself, unlikely to succeed this time with fewer resources when he had failed earlier with more.

December 21 is not only the darkest day of the year but the day on which the darkening ends. For the next six months, the days get longer, not shorter. December 21 is the ancient holiday of renewal. It would have been convenient for this history of the Software Engineering Code of Ethics if the crucial event in SEEPP's renewal had occurred on December 21, 1995. Instead, it occurred one week earlier.

## NOTES

[1] There were actually thirty-two responses by early January, but three are not counted here because the respondents simply asked to be kept informed and did not fill out the standard form (or otherwise provide the information requested). Gotterbarn's first list of volunteers contains only twenty-four names (five less than the total number of respondents even after these subtractions). Most of missing respondents applied before the November 5 deadline, but some did not. It is unclear what the criteria of selection were. Gotterbarn no longer recalls.

[2] One respondent simply sent in an email answering most, but not all, the questions, informally. He is counted in the 29.

[3] Though SEEPP's ratio of academics to practitioners is less lopsided than that of Anderson's committee, it still seemed lopsided to at least one observer: "I was somewhat concerned about the relative lack of practicing computer scientists and software engineers on the task force– it seemed to me that while ethicists could perhaps catalyze and facilitate discussions about software engineering ethics, there needed to be a stronger component coming from actual engineers who faced and ultimately had to deal with the ethical issues that arose in practice." Interview of Steven Barber, April 24, 2002.

[4] Gotterbarn\SEEPP1994-95\Problems with SEEPP 6-8-95.

[5] Gotterbarn\Steering Committee\early essays\Workgrp (2\13\1995). Typical of the ten suggestions is the first:

Our—the seep—schedule has to be revised and made known to the working groups so that they have some type of target. The balloting process will consume a significant amount of time which I don't think we planned for. 4.3.2 has the task force conduct[ing] a ballot—we don't seem to have any provisions for consensus other than the ballot. Shouldn't consensus be achieved in the wg before something gets sent to ballot?

[6] Gotterbarn did not exclude other ways of carrying on the work. After describing the second part of his plan, he returned to the subject of logistics. Though "for the present" meetings will be "electronic", there are "some national meetings that several of us will be at and we can gather in subgroups [there]". Further, if there is a "consensus" of the working group that "we need to discuss something together", he can arrange a conference call (though these are hard to organize, expensive, and therefore to be kept to a minimum). Gotterbarn\SEEPP1994-95\Problems with SEEPP 6-8-95.

[7] Gotterbarn (comments on Chapter 3, June 5, 2003) believes that "the operations guide was of absolutely no help—which part of it did we have to pay attention to ???—Look at the letter again. It says here is what we are doing—my plan and then attaches the 'Guide to operations'." Whether or not SEEPP had to pay any attention to the Guide, there are certainly obvious ways in which it did (or, at least, seems to have). For example, the Guide calls for division of the task force's large project into smaller assignments for working groups. Gotterbarn would later ask his working group to develop a Scope, just as the Guide required. In February 1996 (as we shall see in Chapter 6), he recommended that Keith Miller be "the scribe" to combine the documents expected from the working groups. And so on. Gotterbarn may not have taken the Guide seriously, but he said nothing then or later to indicate that; nor do his later actions so indicate. Gotterbarn is, I think, reading into the past a hard lesson learned only later, that is, that the IEEE Standards procedures, even in Melford's simplified form, could be of no help until someone had prepared a first draft of the code.

[8] Archive, February 22, 1995. The use of "fuses" as an example suggests how much of the document may be IEEE boilerplate.

[9] Phone interview of Patricia Douglas, March 4, 2003.

[10] Gotterbarn\SEEPP 1994-95\Problems with SEEPP (6-8-95).

[11] Gotterbarn archive\SEEPP 1994-95\Problems with SEEPP (6-8-95). Gotterbarn does not recall whether he sent this letter.

[12] My most important publications on codes were then: Michael Davis, "The Moral Authority of a Professional Code", *Authority Revisited: NOMOS XXIX* (New York University Press: New York, 1987), pp. 302-337; and Michael Davis, "Thinking like an Engineer: The Place of a Code of Ethics in the Practice of a Profession", *Philosophy and Public Affairs* 20 (Spring 1991): 150-167. I have published a good deal more on codes since 1993.

[13] Archive, December 12, 1994.

[14] Burnstein received a Ph.D. in chemistry in 1969 from IIT. She also has a masters in computer science from IIT (1984) and a masters in chemistry from the University of Maryland (1965). She was encouraged to enter the field of software engineering by a colleague in chemistry (apparently impressed by her facility in programming). However, her first formal involvement in computer science teaching had another origin. She was volunteering as a "computer mother" in her children's elementary school. She enjoyed working with children and thought teaching them a great opportunity to use the programming she had learned as part of training in chemistry. That experience was one of the factors that got her interested in going back to graduate school in computer science and in teaching in that field. Burnstein had been programming (software) since the 60s. Interview of Ilene Burnstein (May 14, 2002).

[15] The confusion may run deeper. Gotterbarn's archives include a list of 23 under the heading "MEMBERS OF THE INFORMED CONSENT MAILING LIST". Informed Consent is not one of the original eight working group. But among its members are IIT's three (Burnstein, Weil, and me) as well as Gotterbarn, Melford, and Miller—and many others familiar from the lists Gotterbarn drew up of the members of other working groups.

[16] Though the list itself had a title clearly identifying it as the list of the working group, there are two reasons why that clarity may have been lost. One is simply that the format in which the email was printed (all the routing codes) made overlooking the title easy. The other reason the clarity may have been lost is that the email's Subject is "current list of volunteers", not "current list of Professional Competence volunteers".

[17] Archive, June 10, 1995.

[18] A review of the addresses on Gotterbarn's June 6 memos seems to explain why we did not receive those. Only my name is on the list and my address had become corrupted, that is, been mixed up with an ETSU address (CSEP@IITVAX.EAST_TENN_ST.EDU). Explaining why we did not receive the July 1 memo is more difficult. Weil is back on the list (though Burnstein is not) and both her address and mine now seem correct (@CHARLIE.ACC.IIT.EDU). Mechler Archive, E950610, E950610A, and E950701. Mechler had sent his own summary directly to Gotterbarn, not to the list, allowing Gotterbarn to distribute it.

[19] Mechler archive, E951103B.

[20] Mechler archive, E951104.

[21] Mechler archive, E951106A.

[22] Mechler archive, E951108.

[23] Mechler archive, E951114A.

[24] Mechler archive, E951114B.

[25] The Englishman who "hailed from India" would respond on November 22, bringing the total of "live" volunteers (including Mechler himself) to eight: "I also sent a message to chair without any response. When Mario Barbacci was here in September he was hopeful that this exercise would be completed and that volunteers are working on it. I am not sure that we are progressing." Mechler archive, E951122.

[26] Mechler archive, E951109A.

[27] Mechler archive, E951116.

[28] Mechler archive, E951116.

[29] Gotterbarn\Prfcmp-l archive\BOBOUT.ASC

[30] Gotterbarn\Prfcmp-l archive\BOBOUT.ASC. The title of this part of the document is "MESSAGE TO BE SENT TO MY TASK FORCE ON PROFESSIONAL COMPETENCE". Apparently, even Gotterbarn had trouble keeping terms straight: he should have said "WORKING GROUP". But, of course, this is a draft, not a final document.

[31] Gotterbarn must mean "June", not "July". While there is a July 1 email (that IIT did not receive), it does not fit the description (an email calling for work). Apparently, Gotterbarn is working from memory (miscued, perhaps, by Mechler's reference to "July").

[32] Who then were Gotterbarn's "three"? Even if we count Manny Norman as the second, we have a third to account for. One possibility is Mechler. By then, he had done "some work" (as we shall see in Chapter 5). Another possibility is Nayarana Jayaram (the Englishman who hailed from India). As he recalled (interview, February 25, 2003): "I collected British codes and did a comparison….I sent the results to Ed Mechler in Pittsburgh, Amr El-Kadi in Egypt, and some others who were doing something similar. We wanted to see what the state of the art was." While Jayaram's memory places these events two years too early ("at the end of 1993 or the beginning of 1994"), I think it reasonable to add two years to his memory. (After all, he did not volunteer for SEEPP until the *end* of 1994.) Even so, for Jayaram to be the third, we must suppose that one of the "others" to whom he sent his work was Gotterbarn (or that someone else relayed the work to Gotterbarn). Gotterbarn's archives provide no evidence that he received Jayaram's work (or Norman's or anyone else's but El-Kadi's). So, we have a mystery. Indeed, the mystery runs further. Jayaram's work does not seem to have survived at all. It may be another victim of the ETSU list. Gotterbarn has no explanation. Those of us who have lost files as we moved records from one computer to another can sympathize.

[33] Interview of Robert Melford, October 31, 2002. Apparently, it took Melford almost a year to realize the craziness would not soon go away: “I had to discontinue my involvement in late 96, early 97, because my own work had gotten too crazy.”

# Chapter 5: Version 1.0, the Miracle of ’96

“No system is so perfect that it cannot be made to work.”
—Anonymous Engineer

## 5.1 Mechler Who?

Edmund Mechler had worked for several years in the engineering department of a Pittsburgh natural gas utility before he became a SEEPP volunteer. He managed projects concerned with developing special computer systems, geographical information systems, and special document handling systems. He saw SEEPP’s work as an extension of what he was trying to do in his department:

> I basically thought…the software industry [keeps] going around in circles. And at that time they were really going around in circles. The material that they handled is really not that complex. In fact, the computer is really a simple-minded thing. But there’s so many pieces of the simple mind. And we kept going around in circles, almost like reinventing the wheel. And I figured, well,…if they put an engineer behind it and they get the people to start doing that and take it out of the realm [of do it your own way], maybe we can really concentrate on making it an engineering thing that really builds on itself and starts moving ahead instead of just keep going around in circles. I mean, even now [2002] you can pick up a book and it’s a matter of terms. And soon as you figure out what the terms are, you’re right back 10 to 20 years ago. It really doesn’t change. [1]

Mechler seems to have been an ordinary software engineer. While working as an electrical technician at MIT’s Instrumentation Lab (helping to develop hardware and software for research applications), he decided to become an electrical engineer and started taking classes at Northeastern University in 1970. After four years at the Instrumentation Lab, he returned to Pittsburgh, planning to pursue his engineering degree full-time at to the University of Pittsburgh. But, faced with “a lab *four* hours a night [once a week] for *one* credit”, he switched from engineering to mathematics (which, in those days, did not require labs even though it housed computer science). Mechler graduated *cum laude* with a BS in Mathematics in 1976 and, having avoided that four-hour lab, returned to engineering, receiving a master’s degree in industrial engineering from Pittsburgh in 1979. He then worked for industry, primarily in engineering (electrical, computer, or industrial)—at first in hardware and then software. Eventually, Mechler moved into project management, “the communications part of” software engineering and was certified as both a CCP (Certified Computer Professional) and a PMP (Project Management Professional). Though he lived and worked in Pittsburgh during the early years of the Software Engineering Institute, Mechler had had no contact with it when, early in 1995, he first learned of the joint IEEE-CS/ACM project from a trade publication, *IEEE Institute*.[2] He immediately contacted Melford. Melford had IEEE-CS fax him a complete Call (February 6). The final two pages of the Call contained a blank application form. Mechler filled it in and faxed it to Melford. Melford emailed Mechler on March 2, acknowledging receipt of the application and assuring

him that "of course you are a member of the working Group(s) you checked off." He also warned Mechler, "As this is a volunteer effort, our turn-around times tend to lag. You should receive some confirming e-mail in a week or so. If you don't, please let us know!"[3]

Mechler chose SEEPP over the other two task forces because it was, as far he knew, the only one at that time asking for volunteers.[4] A middle-level technical manager in a utility, he was concerned with getting those he worked with to follow standard procedures.[5] "Professional Competence" (Gotterbarn's working group) seemed the category closest to that concern. But, like Gotterbarn, Mechler saw participation in that working group as a step toward certification. As he wrote a friend (March 13, 1995):

> Did you know that the IEEE Computer Society and the Association for Computing Machinery have formed a Joint Steering Committee for the establishment of Software Engineering as a Profession[?] Part of which is certification. I have joined a working group on professional competence and hope to join one on certification. But…the education issues must be solved before certification[,] according to Dr. Mario Barbachi [sic] of the Software Engineering Institute, the head of the Joint Steering Committee [whom Mechler had talked to by phone earlier that day].[6]

Mechler had called Barbacci because he had heard nothing from Gotterbarn's working group and was anxious to do something. He told Barbacci he was willing to "assist in any capacity you may need". He specifically expressed an interest in the following subjects within the Body of Knowledge: Communication; Real Time Systems: Client/Server; Project Management; and Parallel Computers.[7]

On April 21, Mechler wrote Melford again: "I haven't heard from you for a while. When will the committee be in action?"[8] Melford responded four days later: "Apologies for the delay, Ed. You should receive e-mail this week or next."[9] While we do not know whether Mechler got the email (presumably from Gotterbarn) within the next two weeks (as promised), we do know that Melford had informed Gotterbarn of the new volunteer by June—because Mechler received two emails from Gotterbarn on June 10.[10] Gotterbarn had not yet added Mechler to the basic profcomp list; so, what Melford received was what looks like a forward of the profcomp message. The first email laid out a general plan; the second listed some codes of ethics to "look at" and asked volunteers both to choose some to report on and to let Gotterbarn know which they chose.[11] This was just what Mechler had been waiting for. "We started out very easy…let's look at some other codes and see what we can get out of [them]. That's not a bad idea."[12] On June 29, he sent in two summaries: one of the IEEE code; the other for the National Society of Professional Engineers.[13] On July 1, Gotterbarn sent these to profcomp list with encouraging news about what other volunteers were doing:

> We have some volunteers for the second item in the task list [of June 10]. Ed Mechler has jumped in with some engineering codes. I have appended his contribution. Since we are interested primarily in competence, it might be useful to write a little bit about what each code says or fails to say about competence. This will help us get a discussion started. Ed…would you be willing to carry your work this one further step.[14] Joel Fineman is looking at BAR association codes, and Amr El-Kadi has offered to work

> on some of the other codes I listed, e.g., the business communications and the conservation codes. He will also look at the Microsoft Certification Success Guide by John Mueller…I will look at the ACM and British Computer Society Codes. Thanks for being willing to help.
>
> We still need volunteers for the first and third tasks on my initial memo. I am working on setting up a listserv for us. I will let you know. If any of you can do this easily (I have to convince another university [University of Tennessee] to loan us their facilities) it would be great if you could set this up. Let me know.[15]

Some of the other volunteers also did summaries, though the only one to go out through Gotterbarn's list came from El-Kadi (August 31).

Sometime late in the fall (as he recalls), Mechler began to worry—and to act.

> In the beginning, when we were asked to review the models [other codes of ethics] in SEEPP, we were given a deadline for the first copy [of the code]. To a project manager this is very important. When I couldn't get any answer from the emails [to Gotterbarn] I just took over to meet the deadline.[16]

Mechler was not in a good position to "take over". "I knew there was a joint task force [steering committee]. I knew there were three committees [task forces]. How SEEPP was structured, I had no idea."[17] He had no idea how large Gotterbarn's working group was—or, at least, was supposed to be (even though he had the email addresses on Gotterbarn's header). He thought of the working group as consisting of a "half a dozen" people.[18]

## 5.2 Starting from Zero

On December 4, 1995, Mechler emailed that "half a dozen" (with a "cc" to "Barbachi, Ph.D."—because Jayaram had suggested keeping him informed).[19] The half dozen (beside Jayaram) were: (Manny) Norman, El-Kadi, Sullivan, Burnstein, and Weil. (For some reason, my name was not among the half dozen.) That email began much like one of Gotterbarn's. Mechler was writing only to those who had responded to his November email; he had communicated with Gotterbarn separately ("I finally got in touch with Don G. and he reported there were problems with the group: no reply to what yet."); he (Mechler) had a plan for how to proceed. But, having started as Gotterbarn might have, Mechler did something that Gotterbarn had not done. He "note[d] some of the work I have been doing, some additional resources, and a preliminary outline." The "preliminary outline", a page and a half long single-spaced, was something new—in effect, a rough draft of a code of ethics (though also very like the code summaries Mechler had done). It began:

> INTRODUCTION
> SYSTEM DEVELOPMENT
> Evaluate business affects [sic] and culture changes
> Implement Ethical evaluation.
> Assure proper Goals and Objectives

Assure proper Development Methodology
Assure proper Project Management
Assure proper testing, debugging, Case Tools, GUI, etc
Only approve safe and accurate documents
Assure proper privacy, accuracy, property, access and people

There followed similar lists of rules for "PROFESSIONAL PRACTICES (Himself/Herself)" AND "PROFESSIONAL OBLIGATIONS (Others)". Mechler offered this list as a "starting point":

> I agree with Manny [Norman] that we should have reviewed more examples but I did not see any from the membership; if you have an outline please pass it on. I did two and have asked for ACM version plus reviewed some articles so I have developed this prototype. I have tried to organize the subcategories but not all that good, feel free to change. Also, [feel] free to change anything else. Please return comments by the end of the year…. I plan to mix in all your comments and get your approval of the outline by Feb, 1996. Then we can decide what the final version for submission should be like.

Mechler had, in other words, bypassed the various IEEE stages Gotterbarn had regularly set out (before Mechler began receiving SEEPP mailings). Mechler had also ignored the role of the other working groups (from which he had heard nothing). He even ignored the distinction between "code of ethics" and "professional practices" (something else lost in documents "before his time"). He had simply jumped into drafting a document that was to look something like the documents he had been examining. This may seem like the wrong way to write a code of ethics, to work on the upper stories before a proper foundation has been laid, definitely not the way to write a complex software program. But this "starting point" grew into "Version 0" of the Software Engineering Code of Ethics (see 5. Appendix). It was in fact the beginning of what Gotterbarn had called a "straw man".

The effect was immediate. Though I had intended to remain a quiet observer, I responded on December 5 with an email entitled "Additions to your e-mail list of December 4, 1995": [20]

> "Accept responsibility...." How about something about environment? Software, in many uses, may have substantial impact on the environment not clearly covered under "public health, safety, and welfare." I believe civil engineering now has such a provision.
>
> "Assure employees know..." Good. Now, what about employers and supervisors? If a software professional doesn't pro-actively assure that they know the constraints he operates under, they may react with anger rather than understanding to a mid-crisis announcement that "I can't do that because it is against my profession's ethics." For example, certified financial analysts are supposed to give their employers a copy of the CFA code of ethics each year (and must certify to their association that they have done so).
>
> Otherwise, a thorough list. Very good job.

When I wrote these comments, I imagined they would be one set among half a dozen. In fact, they seem to have been the *only* comments. So, on December 14, Mechler wrote his handful of volunteers again (now seven with my name added)—with the uninformative subject heading "Copy of: Continuation – More":

> Volunteers
>
> Here is some additional info from
> The Unwritten Laws of Engineering - W. J. King
> Pa State Registration Board of Professional Engineers,
> Land Surveyors, and Geologists Newsletter
> concerning registration law.
> Michael Davis-Works at the IIT Center for the Study of
> Ethics-Response to Continuations12/4/95
>
> INTRODUCTION
> SYSTEM DEVELOPMENT
> Assure proper estimates
> Assure specifications are fully understood
> PROFESSIONAL PRACTICES (Himself/Herself)
> Add environment issues to "Accept Responsibility .."
> Faithful Agent
> Put forth best effort
> Show initiative on projects
> See through to successful finish
> Don't try to do it all yourself
> Don't ignore signs of trouble
> Don't dodge the issues
> View matters from others points of view
> Accept full responsibility
> PROFESSIONAL OBLIGATIONS (Others)
> Assure employers and supervisors know of code of ethics
> Voice concerns
> Don't supplant another engr after steps have been taken
> for employment
> Fair and just compensation
> Never invade another division's domain without knowledge
> Give fair hearings
> Don't prevent better opportunity elsewhere
>
> I hope to hear from all of you soon.

These new items would be added to the first "straw man". Norman responded the same day with an apology: "I hope to get my stuff about the accounting profession out over the Christmas break. Life is just so busy during work times!" He was as good as his word. On December 28, he sent out his extracts, using Gotterbarn's "list" (rather than Mechler's visible list of addresses). On March 18 (almost three months later), Mechler emailed the "final outline" to his seven (with a cc to "M. Barbachi, Ph.D" even though Barbacci had become IEEE-CS President in January and Felipe Cabrera had replaced him as chair of the Joint Committee):[21]

> Volunteers,
>
> I have not heard from Don Gotterbarn since his e-mail dated 12/15/95; if any of you have heard please let me know. I do not know the conditions of Task Force or Steering Committee; this is why Mario is cc. Well, on with our task.
>
> Below is the final outline; examples are getting too repetitive. It has been updated from Manny's e-mail dated 12/28/95 (please Manny change anything you want) and Project Management Institutes Code of Ethics. I would suggest the next step be: each of us pick a section and add words to make it more readable i.e. Article I: Software Engineering Professionals in System Development
>
> must assure the following factors: (did mine!!!)
>
> Please pick a section or propose another approach and let me know by 3/25/96.
> We will try to complete the task by 4/15/96.

Mechler sounds optimistic in this memo. But a comparison of what he said with what he did *not* say suggests that he had not yet solved Gotterbarn's problem (getting the working group to work). First, he explicitly mentions only Norman's ("Manny's") contribution. We have no evidence of any other (except mine). So, even among Mechler's small group, only three had contributed anything to the code—and, of these, Mechler alone had contributed almost everything.[22]

Second, two days later, Mechler e-mailed "the remainder of our new sub-task force" to announce that he was "assuming a facilitator position, if there are no objections, to bring this task to some degree of completion." He was in effect taking over Gotterbarn's position as head of the Professional Competence working group—except, apparently, he had not yet realized that there was such a working group and thought his sub-group the remnant ("remainder") of the task force (rather than of the working group) . Hence, from then on, he would occasionally use the acronym "SEEPP/E" for his "sub-group".[23]

Having in effect replaced Gotterbarn, Mechler seems to have begun to think in the same orderly way Gotterbarn had. Mechler did not stick to the plan he had announced two days before but returned to (something closer to) Gotterbarn's:

> First we should agree on [what] the final product should look like. I like an intro sentence with bullets. Most of the ones [codes] I review[ed] were like this. Please agree or suggest options by 4/8/96. Then we will divide the sections between the members.

Third, Mechler does not seem to have expected much to come of his suggestion that the format be chosen first—or that "each of us pick a section and add words to make it more readable" (Gotterbarn's "divide and conquer" again). On the same day that he sent off this general email, he emailed me:

> I will be in CHI from 4/1 to 4/3 at the AIIM Conference at McCormick Place. Is it near you? How about lunch?

I accepted the offer two days later, curious to know how much Mechler in person would resemble the Mechler of so many emails. We met on April 2 at my CSEP office and walked the mile to Chinatown for lunch. (Mechler looked like a college professor or senior engineer, average height and build, silvery hair, only a warm smile to distinguish him from a great many other people.) After much sociable back and forth over dim sum, Mechler announced that no one had come forward to "provide the words" for his outline and that he now worried that no one would. The work would come to nothing unless a "Thomas Jefferson" appeared to do for the code what Jefferson had done for the Declaration of Independence. No such person had yet appeared. Mechler was not himself the person; his skills lay elsewhere. The tone of Mechler's voice, the pleading look in his eye, the unfolding of his hands, convinced me that he was asking me to be the code's Jefferson.

I had only one good reason to refuse. I thought of myself as a mere observer (*participant*-observer only in the sense that I was not to stand out as an observer but blend into the crowd of participants). My two small suggestions of December 5 had made me a marginal participant. To do more, to do for the software engineers' code of ethics what Jefferson had done for the Declaration of Independence, was to move from the margins of the process to the center, potentially changing the dynamics in ways hard to predict but certain to invite questions about whether the process was typical. The process (and the code) might have too much of me in it, too little software engineer. Volunteering to draft the code might hopelessly contaminate the process CSEP was supposed to study.

On the other hand, the process seemed about to come to a halt—again. And, given the few volunteers who still were active, this halt might not be so much another halt as the last, final, and disastrous end of the project. To fail to help draft the code might well mean there would be no drafting to study. An unusual opportunity would be lost. In this moment of crisis, the risk of contamination seemed worth taking.

I did not know why Mechler had picked me (except that I had showed interest and apparently could write), but I did think I could do a good job.[24] Though a philosopher (Ph.D., University of Michigan, 1972), I had had a year of law school (1976-77, also at the UM, when I could not find a job in philosophy). Among the usual (required) first-year courses, there had been one elective. I had chosen comparative law and learned much about French and German legal codes, including the experimentation and theorizing that preceded them. I had been especially impressed by the way the major nineteenth century legal codes combined "general clauses" (such as "An immoral transaction is void") with more specific rules ("Where the subject of a shareholders' resolution is the appointment of auditors for the examination of certain events connected with the organization or management of the corporation, those shareholders who are directors or officers of the corporation may not vote"). The summer after my first year of law

school, I clerked for the Secretary of the Michigan Law Revision Commission (a UM faculty member). The subject of the laws he helped to revise that summer was special assessments, a subject only a real estate lawyer could get passionate about. But the problem of putting more than a dozen independent statutes into a single, sensibly organized, easy to use, and politically acceptable form was not so different from the problem Mechler's list now posed.

I also had an interest in professional codes going back to the 1974-75 academic year, when I had taught legal ethics at the law school of Case-Western Reserve University. Lawyers were just learning to use the new *Code of Professional Responsibility* (1970) after abandoning the old *Canons of Ethics* (originally adopted in 1908 but much amended since). The Canons were a simple list of *do*'s and *don't*'s, not so different from Mechler's Version 0. The new Code was divided into seven Canons (general provisions). Under each Canon were several specific Disciplinary Rules (enforceable) and Ethical Considerations (the exact force of which was a subject of controversy even though the Code's "Preamble and Preliminary Statement" described them as "aspirational"). Because this three-level structure did not correspond to any other document with which lawyers generally worked, there had been much discussion among them about what codes should do and how they should do it, a discussion that led to the abandonment of the 1970 Code in 1984. The code that replaced it (*The Model Rules of Professional Conduct*) had the standard format of a model statute, that is, mandatory rules with commentary to explain how to interpret the rules. Any "aspirations" went into the commentary as guides to interpretation.

I had begun writing about professional codes in the mid-1980s.[25] I had begun studying engineering codes about the same time.[26] I had even written a theoretical piece with part of a model code in it (though one concerned with governmental ethics).[27] I felt that I had been preparing to be some professional code's Jefferson for a long time. While I knew that I knew too little about software engineering to write its code alone, I thought I could count on Mechler, my colleagues at IIT (especially Burnstein), and perhaps some other volunteers to compensate for my ignorance.

I should, however, not then have made the decision to draft the code. Even though the NSF grant that had brought me to lunch that day had explicitly treated providing technical assistance as one of the purposes of the grant, the decision to provide that assistance belonged to the research group, not to me alone. I should have told Mechler that I was inclined to volunteer, if Mechler would have me, but that I could not volunteer unless the research group agreed. Mechler would no doubt have understood—and waited. Instead, I immediately said I would try to do a draft, perhaps hinting that I would have to consult Weil to be sure I could do it. Mechler's response was enthusiastic. He left no doubt that he had hoped I would volunteer.[28]

We should not conclude from these events that Mechler scheduled the lunch in order to recruit me as "wordsmith" (or, at least, not with that purpose alone). Mechler had just had a lesson in the importance of visiting with members of the working group when passing through town. On March 18, he had received an email from Jayaram—which read in part:

> Regarding the elusive Don Gotterbarn, you are singularly lucky having got his email [of December 15, 1995]. He was in LONDON during Feb 2nd of this year presenting his paper—you guessed it—on ethical issues in SE! This Conf titled "Professional Issues in SE" was organized by my colleague (who shares the office

> space with me!). All that was required of Don—if he was remotely interested—was to give me a call spending as little as a quarter or ask my colleague who was a co-chair or take a brisk walk worth 15 minutes from the venue or use his subway ticket purchased for the day for a 5 minute ride TO GET ME AT MY UNIVERSITY (I had to take my graduate-level classes that day)!

While this e-mail also complains of Gotterbarn's failure to respond to e-mail (a subject to which we will return in Chapter 6), its heat suggests that no amount of email could substitute for the courtesy of a personal visit. Gotterbarn did not visit Jayaram because, though attending the "Westminster PASE'96 Conference London", he did not realize that the "@Westminster" in Jayaram's email address meant that, like the conference, Jayaram was located in that part of London (once an independent city) in which Westminster Abbey is located.[29] Gotterbarn must by then also have forgotten the list of working group members he had last updated nine months before; Jayaram's address there is "London".[30]

Jayaram knew that Gotterbarn was at the conference because Gotterbarn gave a paper (actually, two) and so had his name listed in the conference program. But Melford's name also appeared in the program; he too gave a paper. Melford also did not look Jayaram up.[31] Why did Jayaram not also complain about Melford's failure to pay a courtesy call? There are two possible explanations. One is that Jayaram was thinking of Gotterbarn as leader of Jayaram's working group—and was complaining about his conduct in that capacity (rather than in his capacity as SEEPP co-chair). That Melford was the other SEEPP co-chair would then be irrelevant. The other possible explanation is that Melford's name no longer meant anything to Jayaram. Melford was by then invisible to most members of SEEPP's working groups (or, at least, of Gotterbarn's working group). For a long time now, the only messages from SEEPP (or any of its working groups) had come from Gotterbarn. Of these two (compatible) explanations, the second seems the more likely. Like many other members of Gotterbarn's working group, Jayaram may have lost track of the distinction between SEEPP and the Professional Competence Working Group. For him, the working group was SEEPP and SEEPP was Gotterbarn.

Mechler naturally assumed that Gotterbarn meant no discourtesy and decided to prevent such an oversight from weakening his "sub-group". So, on March 19, he told Jayaram that he had taken "the liberty to send [your reply] to the other team members working on the ethics outline". Then, on April 5 (after his meeting with me) he emailed the sub-group: "Just in case any of us travel I would like to list what city each of you lives in. If one of us travels to a member's city we could meet." He then listed the eight members of the group (including himself and me), asking each to email his or her location. On April 10, he emailed to all a list with location for each. (Plainly, the five who had had nothing to say about Mechler's "straw man" were still reading their email and anxious to stay involved: they had all sent in a home town.)[32]

## 5.3 "Mr. Jefferson" at Work

A few days after my meeting with Mechler, I informed Weil that I had agreed to do a first draft of the code.[33] She approved. She had missed the meeting only because she was out of town. She had already tried to help Mechler get on with the writing—without herself consulting the research group. On March 25, she had written: "I am glad to see the work going forward. Your

list of March 18 is quite comprehensive. Do you plan to put the imperatives in categories with rationale or preamble to each category?"[34] Mechler responded the next day that he was "open to discussion. Maybe give an example of one of the categories. I wasn't myself but each member can take a part or suggest a form." Weil had concluded from this exchange (especially, from the request that she give examples) that Mechler had done all he could and that the group as a whole would not do more. She had herself begun to think of who might be brought in to do the writing. She had come to the same conclusion as Mechler.

During our April meeting in Chinatown, Mechler and I had discussed schedule. I had pointed out that I was in the last weeks of the semester. I could not do much before May. On May 2, Mechler emailed a reminder: "During our meet in Chi. you said that you would add the 'good words' to the ethics outline. Was the text copy I sent cc good enough for this task? Let me know if you want a paper/WORD/WordPerfect/etc. copy." I responded (May 8): "WordPerfect copy would help. Probably cannot finish job before June 15. O.K."[35] On May 15, Mechler emailed "With regard to adding words to the outline, I am sending you a disc with three versions [of WordPerfect]. Let me know if one is ok. Also have dos and secondary file if you want. June 15 ok for finish date." The diskette arrived about the same time as the email. I set to work immediately.[36]

By May 24, I had a rough draft—and several problems on which I wanted Mechler's advice before I went further. I wanted Mechler's advice for three reasons:

> First, it turns out that there are some clauses which, when I started trying to put into prose, I realized I didn't understand. (See ?'s.) Second, there are some clauses I didn't know what to do with, either because they didn't seem to belong in an ethics code (too specific), because they seemed to repeat what I already had, or because I couldn't figure out where to put them. (Hence, the unnumbered clusters at the end of some sections.) Third, I have found myself rearranging your list. Have I missed something I should have seen; or have I really made an improvement?

Because I had also changed Mechler's (computer) formatting, I "didn't dare to send" the draft email. I therefore asked for Mechler's fax number.

Mechler took four days to respond. His four paragraphs suggest that he had a lot to think through before responding. He declined to let me pass the buck: "mainly you must decide how far you want to go adding words." He then acknowledged an issue that I had not raised in my May 24 email: "From our conversation in Chi., I believe that you did not want to go too far due to your two roles. In fact, while reading your e-mail a question occurred; when I pass this around to the other members of the group do you want credit or should I say an anonymous member added words?" He then returned to my questions, offering a few options but, essentially, giving permission to do whatever I thought best:

> In regard to your specific questions, they all seem reasonable but should I or the group answer? There is no doubt in my mind that you had to rearrange clauses, found duplicate clauses and ones that are hard to place, if they can be placed. Also, how far should you go? Maybe the effort you have already completed is enough and I [should] pass around

> the results plus questions? Let the group finish the effort. I must leave these questions up to you because of our conversation in Chi.

Recalling that what Mechler had said "in Chi" was that he had gone as far as he could, that there was no one else to do it, and that unless I did it, it would probably not be done, I replied on May 30: "Your email of May 28 gave me courage to do one more draft (if you find the document I sent you to be basically what you had in mind)."

Most of my email is, however, about how to present the draft, once finished, to SEEPP/E. I thought that Mechler, not I, should send it out in part "for practical reasons". In the past, CSEP had had trouble emailing large documents. CSEP's librarian, "the email expert", had just resigned, moved to Denver, and not yet been replaced. But, in part, I thought that Mechler should send out the draft because it was "important that at least one software engineer is happy enough with the document to put it out." That Mechler had to "put it out" did not mean the draft need be anonymous, only that Mechler "should make it clear that I am trying to work within the list of do's and don't's the group developed, providing a framework rather than new substance." I concluded by promising one more draft if Mechler's comments "are mostly favorable."

Mechler took almost a week to respond (in part because my email arrived late on Thursday, after he had left work, and he did not return to his office until Monday). But his response, when it came (June 6), was certainly favorable: "I just read your draft and have only one word for it EXCELLENT EXCELLENT EXCELLENT EXCELLENT. It is a great step forward." He then suggested that "when you find a statement that is unclear or you do not know if it fits put 2 or ??? to highlight the condition." Just to be sure I would not take this minor criticism too hard, Mechler concluded the email as he had begun, with praise: "Really looking forward to the review draft. Again an excellent job, just what I wanted."

My next communication with Mechler is a "classic mail" letter with the same date as Mechler's email (June 6). The letter seems to begin with an (understated) acknowledgment of Mechler's approval: "I took your comments to mean continue. So, I did." Enclosed with the letter was "your disk with the draft code of ethics". Mechler was to "note that this is (more or less) on time and (certainly) within budget." The letter concludes with me looking forward "to your response, but especially to the responses of all those connected to us only by email."

A closer reading of my letter shows that it is in fact *not* a response to Mechler's email of that day. Mechler's letter was received at "15:50:41", rather late in the afternoon on Thursday. How could I revise the code as instructed, put it on disk, and write my covering letter, in what little remained of the working day? Memory does not answer that question, but there are at least three reasons why it is improbable that I rushed out a response. First, I had no reason to rush. I had originally given myself till June 15—and Mechler gave no indication of holding me even to that date. I had more than a week until that target date and, given SEEPP's history until then, a week or two beyond that date should have bothered no one. Second, I generally worked on the code at home because I considered such writing to require concentration. The office was for work that would not suffer from frequent interruption. Third, there is an *email* that seems more like a response to Mechler's. That email was sent on Monday, June 10. In it, I inform Mechler that a diskette "(with a somewhat revised version of the code you saw) is in mail. Please put in '???' as you see fit (all unnumbered phrases)." Apparently, I did not receive Mechler's Thursday email until Monday. (I generally did not come to the office on Friday during the summer.) I had

worked on the code without Mechler's four *EXCELLENT*'s; hence, I could not follow Mechler's suggestion about the use of "???" and had to have Mechler add them himself.[37] That I sent Mechler the revised version on the same day that Mechler's email arrived with its praise, seems to be pure coincidence.

So, what were the comments Mechler had made that encouraged me to do one more draft? They should not, it seems, be those of May 28, since I said I would do one more draft *if* Mechler's comments on the draft on which Mechler had not yet commented were "mostly favorable". Mechler's June 6 comments were the comments I was supposedly waiting for before doing another draft. Did Mechler send any comments between May 28 and June 6? The archives contain none. No one recalls any. And the tone (and substance) of Mechler's June 6 email makes it doubtful that there was any communication in the intervening week. So, it seems, I had not waited for Mechler to comment. I had taken Mechler's May 28 email as permission to continue with detailed revisions even as Mechler assessed the overall document ("mainly you must decide"). I expected Mechler to approve and acted accordingly. What I did not expect was that Mechler's approval would be so unequivocal.

54 The Architecture of Version 1

Then, for an entire month, nothing happened. But, on July 10, the following email from Mechler arrived at CSEP:

> Volunteers,
>
> One of the members of SEEPP/E, Mike Davis, has taken our outline and added the good words. The attached, in text mode, is the result. I think I talk for the entire group that has been working on this when I thank him very very much. Now it is your turn. Please review and send me the changes by the end of July. We should be able to submit the FINAL product by the end of the summer.
>
> As you read each rule or at the end of a rule you will see statements with ??? attached. These were statements that Mike wasn't sure of, thought they were covered, or didn't know where to place. If you think they should be placed tell me where or they will just disappear.
>
> We are finally in site of the end.
>
> Ed

There was no attachment, but the promised document arrived a minute later as a separate email, just over four single-spaced pages long. Version 0 was now very close to what would soon be Version 1 (6. Appendix). There was a preamble (Introduction), seven general clauses (Rules) such as "Software engineers shall, insofar as possible, assure that the software on which they work is useful to public, employer, and user, completed on time and at reasonable cost, and free of significant error" and, under each general clause, between three and eleven more specific

clauses preceded by an "In particular, software engineers shall…" Each Rule had both a number (1-7) and a name in capitals (PRODUCT, PUBLIC, JUDGMENT, CLIENT AND EMPLOYER, PROFESSION, COLLEAGUES, AND SELF). Each specific clause had a decimal number indicating its place under the general clause as a sub-rule. For example, the first specific clause under Rule 1 was 1.01 ("Assure that they understand fully the specifications for software on which they work"). The two decimal places allowed for the number of clauses under a Rule to grow beyond 9. At the end of all but section 3, 5 and 7, there were one or more (left-over) rules or phrases followed by "???". The first of these, for example, was "Assure proper privacy, accuracy".

Given Mechler's favorable response, it is worth noting that I had not merely "added words". I had entirely reworked Version 0, keeping only a) the terms "Introduction" for "preamble" and "Product Development" (shortened to "Product") and b) the wording of provisions I did not understand or thought redundant or inappropriate. Mechler's other categories (and titles) had been abandoned and all the individual rules had been rewritten. Mechler had said he would like a code consisting of an "an intro sentence with bullets. Most of the ones [codes] I review[ed] were like this". The code he got had a more complicated structure. There were no bullets. Instead, I had (it seemed) followed the typical IEEE standard.[38] Each sentence bore a number, making reference easy. The Rules were not simply introductory sentences; they were the general clauses governing interpretation of the more particular rules under the general rule and serving as a backstop when no particular rule applied. My chief innovation, the use of "In particular" before each list of particular rules, was intended to stress that the particular rules were not all that a software engineer need be concerned about. The general rule had to be considered too.

My structure was no closer to that Weil had suggested ("put the imperatives in categories with rationale or preamble to each category"). The general Rules, though serving to categorize the more specific rules, were not mere rationales or preambles. They were (as the code called them) Rules. In this respect, as in some others, my document most resembled codes with which most software engineers would have been familiar. In general structure (and many provisions), it resembled the IEEE's Code of Ethics of 1979, the Code of Ethics of the National Society of Professional Engineers, and ABET's Code of Ethics.

Significantly (as it turned out), my (version of the) code did not much resemble either IEEE's 1990 code of ethics or (except for its decimal numbering) the ACM's 1991 code. The IEEE had drastically shortened its 1979 code to a brief preamble and ten imperatives (such as "1. to accept responsibility in making engineering decisions consistent with the safety, health and welfare of the public, and to disclose promptly factors that might endanger the public or the environment"). The ACM's 1991 Code of Ethics and Professional Conduct, though a long code like mine, was organized in a different way: an unusually long (six-paragraph) preamble, twenty-four "imperatives" (a little over a page long), and "Guidelines" inserted between the imperatives to provide one or more paragraphs of guidance on interpretation of the imperative immediately above. The imperatives are divided into four categories: General Moral Imperatives (such as "1.1 Contribute to society and human well-being"), More Specific Professional Responsibilities ("2.1 Strive to achieve the highest quality, effectiveness, and dignity in both the process and products of professional work"), Organizational Leadership Imperatives ("3.1 Articulate social responsibilities of members of an organizational unit and encourage full acceptance of those

responsibilities"), and Compliance with the Code ("4.1 Uphold and promote the principles of this Code"). My code had many more imperatives than the ACM code had; it did not distinguish between (what the ACM's Preamble called) "fundamental ethical considerations" and "more specific considerations of professional conduct" (though it had a similar distinction between general and specific professional rules). My code did not make a point of the professional's "leadership role". In these ways at least, my code could easily seem to anyone familiar with the current codes of the two sponsoring organizations to be a retrograde development, a throwback to a style of code deliberately abandoned for something better.

We might then have expected Mechler's reaction to have been more ambivalent than it was. Certainly, given the diversity of Mechler's little group, we would expect at least one of its members to question the code's structure. None did. SEEPP/E's reaction was as favorable as Mechler's. We can almost hear the sigh of relief. At last, here is a draft code with which they could work. The structure was familiar enough that it did not get in the way of what seemed important: the particular requirements. One of the four SEEPP/E members who had had nothing to add to Mechler's list now responded in detail. That response (July 14), more than a half page long, came from Egypt (through Gotterbarn's PRFCMP list). Among El-Kadi suggestions were removing "3.06. Decline to contract for a service with self. (?) ???", replacing the word "badly" in 4.06 with "not as excepted" or "inappropriate", and renumbering (and moving) 5.04 to 6.04. Norman's response, sent only to Mechler (July 20), was similar—with one exception—, though more than two pages long: Norman proposed one new rule: 7.04 "Endeavor to keep current with advances and changes in the field."

## 5.5 Drafts, Drafters, and Draftees

Early in August, Mechler concluded that he had received all the comments he would receive (just two). He therefore emailed his "Volunteers" (August 8, 1996) the following (with a cc to Cabrera):

> Attached, in text form, is the final product of the SEEPP/E subgroup after adding your comments and eliminating ??? that were not necessary. If you have any outstanding additions etc., please let me know by the end of August because at that time I will submit it as our final product.

This email brought a surprising response, a long fax (eight pages, single spaced). On August 17, I wrote:

> I guess I misunderstood your memo of July 20. I didn't realize you intended to do the final draft. I thought I would get the suggestions and incorporate them. I think this would be better for two reasons: 1) to keep the single style (a minor matter); and 2) to take into account suggestions Bernstein [sic] and Weil have made—and with which I agree (a more significant matter). I also had left some loose ends I wanted to clean up. I guess I should have put all this on email—like everyone else—but I didn't. So, I'm now doing this fax to give you some idea of thinking here.

Following this introductory paragraph, there was a second explaining the two-and-half pages of annotations on the document that formed the second half of the fax. That document was *not* Mechler's "final code" but the one I had prepared largely independently, scavenging Mechler's "final code" when it arrived. I downplayed the differences. Except for "the change of an 'its' to a 'the' in the last line of the introduction" and the addition of "endeavor" at the end of Rule 7, all changes were (I noted) in the subsidiary clauses and "even these changes leave more than 90% of the text unchanged." Yet, whatever the similarities, the two codes did differ significantly—as the two pages of annotations comparing the "M" code (Mechler's) with the "D" code (Davis') made clear. What had happened?

As so often during SEEPP's early years, there had been a failure of communication. I had just assumed that I would remain in control of the drafting. Though I had *suggested* that I would do *only* "one more draft", I had not actually said that I would do only one more—and I had not meant that I would do only one more. What I had meant was that I was ready to withdraw if Mechler thought I was getting in the way of software engineers doing what they wanted. But so far, the software engineers involved seemed to be happy with what I was doing. And I was enjoying myself immensely. I therefore had no reason to withdraw and so had not. I had, however, said nothing about this to Mechler. From what I had said, Mechler drew the obvious inference that he was lucky to get as much out of me as he had. Since he had a document with which he could work, he had worked with it, accepting Norman's and El-Kadi' suggestions whenever he could.

Mechler also had not guessed that I was using IIT's research group to provide the sort of communal deliberation on the code that email had not provided. Since award of the grant in early 1995, the research group—(primarily) Burnstein, Weil, and I—had been meeting almost monthly to go over SEEPP email, trying to understand the process. But, starting in June, once I had sent Mechler the first draft, Weil had begun convening meetings of the research group every week or two to go over the code itself. Each of the three members of the group seemed to come with a different purpose. Weil seemed primarily interested in understanding the code. Her questions were generally of the what-does-this-mean variety. When neither Burnstein nor I could answer the question, there followed a discussion of what the provision *should* mean. Once there was agreement on that, I revised accordingly. Burnstein, the only software engineer present, was concerned *both* to demand enough of software engineers to raise standards to a point she considered decent—*and* to avoid demanding so much that the code could not generally be followed. She could often state a problem with a provision (or the absence of a provision) without being able to offer appropriate language to fix it. She had remained silent online in part because she was participating in the research group's deliberations but in part because she could not on her own have offered "the words" and did not feel comfortable just complaining. Having worked alone on the code for almost two months, I was happy to have a responsive audience. I would take notes at the meeting, revise the code, and circulate a new draft the following week. What satisfied my two colleagues stayed; what did not was revised until it did. I had a resource Mechler lacked.

Most of the IIT changes in the "D draft" simply clarified an existing clause or moved it. But five added new rules. Three of the new rules originated with Burnstein. Two of these were under Rule 1 (with older clauses renumbered accordingly): "1.01. Assure that specifications for software on which they work have been put in writing, satisfy the user's requirements, and have

the customer's approval" and "1.02. Assure that they understand fully the specifications for software on which they work." The other clause was 7.05 ("Improve their knowledge of the law governing the software and related documents on which they work"). The other two new rules were an attempt to explicate the injunction against "self-dealing" (rather than, as El-Kadi had suggested, simply dropping it): "3.05. Neither solicit nor accept a contract from a governmental body on which a principal or officer of their employer serves as a member" and "3.06. Participate in no decision of a governmental or professional body, as a member or advisor, concerned with software, or related documents, in which they, their employer, or their client have a financial interest." I had found the inspiration for this explication of "self-dealing" in part in the NSPE's code of ethics (II.4.d.) and in part from studying of *ASME v Hydrolevel* (a case in which a volunteer participant in the standards writing process of the American Society of Mechanical Engineers seemed to have written standards designed to eliminate one of his employer's competitors).[39]

In this way, two more of the "silent four" (Burnstein and Weil) made substantial contributions to the code. Only Jayaram and Sullivan did not. Why? One explanation is that they resembled Burnstein in not being able to work alone. They might have made contributions similar to hers had they been able to participate in the sort of face-to-face meetings IIT's research group had. They both think one or more face-to-face meetings would have helped the process.[40] Another possibility is that they saw others doing the work they would have done. They saw no reason to intervene. Both were evidently reading their email; they had responded, for example, as late as April when Mechler had asked for home town. Each had contributed his share to the early work, Sullivan by attending many of SEEPP's early meetings, and Jayaram by summarizing several British codes of ethics (though that contribution seems to have been lost in the ether). Jayaram certainly had said enough when things seemed to be on the wrong track (or when he thought Mechler needed to know who chaired the Joint Steering Committee). Now that the group seemed to be on the right track, Jayaram could read his email with delight.

## 5.6 A Big Mistake

On August 20, 1996, I sent Mechler a diskette version of the code. The covering letter indicated that, while the code was identical to the one faxed on August 18, "you should have received a fax with this revision of M [Mechler's] 4.09, which I suggest [as] new 1.11, renumbering accordingly". This new 1.11 ("Assure that raw information used in software is accurate, derives from a legitimate source, and is used only in ways properly authorized") would eventually prove controversial (as would most of the other late additions).[41]

On September 11, Mechler emailed what was now *almost* Version 1, with a covering note two paragraphs long. After the salutation "Volunteers", he reports:

> We had some last minute suggestions; I have added and attached, in text, is a copy for your review. Please do as fast as possible; if I do not hear from you by Sept. 30th, I will submit as our final product. Please remember each style is different.

Though this email is addressed to Mechler's usual seven, the cc includes not only Cabrera (as was now Mechler's custom) but Gotterbarn (which was reasonable, since Mechler was

technically working under him, but had not been Mechler's custom) *and* Keith Miller—a novelty requiring explanation.

Mechler's explanation, given in the second paragraph, is: "Keith, Don forwarded a copy of your 'Testing, Reliability, and Trustworthiness' to me. After reading, I thought you may want a copy of our 'Ethics'." Miller's working group (Reliability and Safety) seemed to have been working more or less in parallel with SEEPP/E and had (it seemed) sent out its draft (two and a half pages single-spaced) on July 15 (with a revised version coming a day later)—denying Professional Competence (or SEEPP/E) the honor of reporting first. We will tell the story of Miller's document in Chapter 6.3. For now, what is important is Mechler's response. Miller's document arrived just as Mechler was completing the draft he would send out on August 8 (what I labeled "M"). Mechler had time to read, reflect on, and benefit from Miller's document. Mechler nonetheless acknowledges no debt to Miller. Miller's document suggested no revision in what would soon be Version 1. Mechler's only response to Miller's document seems to have been to send his own as a courtesy (as if he were leading a coordinate working group) or perhaps as an expression of pride (since he received Miller's document only because Gotterbarn had forwarded it). While Mechler had to receive his copy through Gotterbarn, each member of IIT's research group had received an individual email copy directly from Miller. Miller must have recalled that we were supposed to be observing the process as part of his working group. I used the document as a check against the code I was working on; I was pleased to find no reason to change what I had done. I did not wonder why I had no emails concerning its drafting.[42]

I sent Mechler two more (minor) changes on September 13. On September 30, 1996, Mechler emailed Cabrera and Gotterbarn (with cc to each of SEEPP/E's seven members): "Attached, in text format, for your review, is the final product of the SEEPP Ethics group. If there is additional work on this product, or any other part of the project, please contact us." Cabrera responded the next day with, "Thank you Ed."[43] On October 4, Mechler took Cabrera's courtesy response as an excuse to underscore SEEPP/E's willingness to remain involved in work on the code (and to make sure a change of email address did not exclude him):

> Felipe
> Thanks for the "Thanks e-mail" and CC to the group. Again if additional work is required we are available. Also, the product is available in other formats if you require a better version; it is held now in "Word".
>
> I have a new e-mail address
> emechler@eri.eqt.com
> I will keep open the CompuServe one for the project for a while. The new one is under Exchange and as you know this is much better, except I need to get a book for various areas I can not get an answer for: too far ahead of total implementation.
>
> What is the status of the Joint Steering Committee and three task forces? When I read about this endeavor, it became very important to me. We really need to move in the direction proposed. Will the results be published? If you need any help, in any area, please contact.

Ed

The same day this friendly note was sent, Dennis Frailey (the Joint Steering Committee's vice chair) emailed the (non ex-officio) members of the committee his comments on the code (with Gotterbarn alone receiving a cc). Two days later (October 6), Cabrera accepted Mechler's offer of help:

> Ed,
>
> Could you address the comments made below [Frailey's email without identification of author]?
>
> Some are tough and general, and perhaps they cannot be addressed. Yet the more pointed ones would be nice to clarify and fix.
>
> Please let me know.
>
> —Felipe

Frailey's second paragraph set the tone of his response: "In general, I think this expresses some good ideas, but is entirely impractical in a number of cases. The problem is that many of the things being proposed are vaguely defined, impossible to accomplish, or not under the control of the software engineer. A few others might not be generally accepted as the 'right' thing to do."[44] Frailey had objections to every item under Rule 1. After that, his criticisms were more scattered but still substantial. He seemed to have gone over the code with some care.

The next day (October 7) Mechler received another email from Cabrera. It began by urging Mechler "not to feel overwhelmed or disappointed but pleased that smart people are reading your document in a careful manner." It then asked him "to address" the points made in the email "below". Below was a forward from Mary Shaw, though Mechler could not tell the source from the version Cabrera sent him. Cabrera had cleansed it of (almost) all references to members of the Steering Committee. Shaw too was unhappy: "I share xx's general concern that this is fine in theory but hard to apply. I realize that codes of ethics should be idealistic, but they also must be credible—that is, practitioners should be able to see how the code shapes their work. Another way of saying this is that this draft seems a little naïve."[45]

The attempt to preserve confidentiality failed. Mechler soon learned the identity of both critics. But the knowledge was of little use. The two critics had votes on the Steering Committee. Though only one-sixth of the Committee's voting members, they might well be speaking for a larger number, enough perhaps to defeat the code if it were put to an immediate vote. Mechler had no idea how the others would vote. He had no choice but to respond to their criticism.

## 5.7 Mechler fights back

Until now, Mechler must have seemed to everyone in SEEPP/E a gentle administrator, one who inevitably worked by consensus, without strong ideas of his own. He had known how to bring a difficult undertaking to completion, but he had shown no deep understanding of what SEEPP/E was doing. He had not objected to anything any SEEPP/E member had proposed. His practice seemed to be to accommodate everyone. Even when I, however unintentionally,

proposed to discard a draft code to which Mechler had devoted considerable effort, about which he must have had some authorial pride, and to which he had publicly committed himself, he had simply accepted my version of the code, letting his own die without a word of objection.[46] Mechler seemed ill-fitted for debate with two stars of computing, one a professor and the other a senior researcher (and adjunct professor). What followed shows how hard it is to read character.

Mechler probably should have taken time to allow his first thoughts to mature into a strategy. He should have consulted others, gathered evidence where that seemed appropriate, and organized his response into a systematic analysis of the criticism, something academics could recognize as a decisive refutation. But email invites quick response (especially to critics who themselves seem to have responded quickly). Mechler did as invited. Looking back with the advantage of half a decade, Mechler admits that "[w]hen I saw the comments I saw red. Not because people were commenting on the product, I expected that. It was the comments themselves." They were like the comments he heard when he tried to introduce his company to processes successful outside.[47]

Mechler responded to Frailey's email on October 6 (the same day he received it), going through it item by item, inserting what he had to say to a criticism right after the criticism. He did the same for Shaw's on October 7. Frailey and Shaw had four distinct kinds of objection in common. Much of the code was either: 1) too idealistic (impractical for anyone), 2) too vague, 3) impossible for a software engineer to accomplish (because not under her control), or 4) not what some would think should be accomplished. To these, Shaw added one criticism of her own (a criticism later revisions of the code would struggle to set to rest): 5) a failure to "discriminate in intensity". She also expressed a general distaste for the code's structure: "On the whole, I think the document would be more likely to be read if it were shorter, crisper, and more realistic. A supplementary document could work through the examples in a little more detail than the 2-liners that constitute most of this document."[48] What she, one of the ACM-appointed members of the Steering Committee, seems to have in mind is, of course, the structure of the ACM's new code of ethics (with its "crisp" two pages of rules and the longer Guidelines with their "examples in a little more detail"). Mechler, an engineer, had not yet seen the ACM code. He therefore did not see what Shaw was getting at. Since his own original conception of the code was closer to Shaw's "crisp" short code, he could only respond with a shrug: "This is a matter of style. We decided on the [NS]PE Code model to list some of the detail in items associated with the rule."[49]

Mechler's response to the other criticisms that Frailey and Shaw made was more pointed. The critics were simply wrong on all five counts. So, for example, in response to the first criticism (good ideals but impractical), Mechler noted that "the SE Code of Ethics was built using other professional codes as models. Ethics doesn't have a practical/impractical side that I have ever experienced." Mechler agreed with Frailey and Shaw that a code of ethics should be practical, not a mere "ideal" (or aspiration). He disagreed only concerning the practicality of Version 1. Mechler's method had (generally) been to go through other codes looking for ideas. Provisions developed in that way could not, Mechler supposed, be impractical insofar as they were (more or less) the same as provisions in other codes that had proved themselves in practice (as evidenced by their long presence in those codes). Of course, in arguing this way, Mechler implicitly undertook to demonstrate that the code as a whole, or at least the particular provisions Frailey or Shaw criticized, were indeed in other codes—or had otherwise proved themselves in

practice. This he did not do.[50] Neither Frailey nor Shaw was likely to take a different view of the practicality of those provisions without such an explicit demonstration.

Concerning the vagueness of provisions (the second criticism), Mechler had two responses. One was simply to deny that a provision was vague. So, for example, Frailey complained that "the last sentence of the introduction makes no sense to me: 'Each subsidiary clause is a specific application of its general rule, one experience has shown needs express statement, but no set of subsidiary clauses exhausts the general rule.'"[51] Mechler's response to Frailey's incomprehension is to offer (what he considers) the obvious gloss: "General disclaimer saying we cannot give all instances of a rule in ethics." Mechler's other response is more telling because it is more general: "Usually, ethics are vaguely defined." Every code must leave much room for interpretation. Interpretation is inevitable. Objection to Version 1 on grounds of mere "vagueness" (as opposed to a specific failure to say something more clearly) is in fact an objection to any code of ethics, no matter how long.

Perhaps the heart of the case Frailey and Shaw made against Version 1 was that many of its provisions are impractical for *software engineers* (however practical for other professionals). In some cases, the problem is that the software engineer simply does not have enough control of the process to do what the code asks; in other cases, the problem is that the cost of doing it is too high; and in other cases yet, there is no point to doing it. Frailey and Shaw can be quite specific about the impracticalities. So, for example, Frailey objects to 1.01 ("Assure that specifications for software on which they work have been put in writing, satisfy the user's requirements, and have the customer's approval") by asking us to suppose "you are brought in to fix something where you have (no) interface with the customer and do not know if the customer has approved." Mechler's specific response is that "[the] example given is one of the problems we are trying to decrease the occurrence of." (This response must be read in the context of his general claim that "I don't think that any of the items are impossible to do as a professional.") His point seems to be that all the rules under Rule 1 are introduced with "In particular, all software engineers shall, *as appropriate*" do as the subsidiary rules require. The code does not actually say that the software engineer in question *must* assure these things even when assuring them is impractical. Frailey's example may well illustrate one of those situations where seeking to do what 1.01 asks *is* inappropriate (because the software engineer is in no position to do it). But even here, it would be good for the software engineer to at least try to do as 1.01 asks (for example, by asking for the specifications or for a way to contact the user or customer). Even when unavoidable, repairing software without knowing what the user needs and the customer wants is never *good* practice. "We" would like to have less of that.

Mechler's response to the fourth criticism is somewhat different. He begins by acknowledging the general point in a way that diffuses it: "I'll give 100 to 1 odds that under any professional code there are items that some will think are not the 'right' thing to do." Codes of ethics generally contain some provisions that "some" think not right. To criticize Version 1 for that is, therefore, no criticism at all, just a fact about writing a code of ethics. The question is not what some might "think" but what *is* "right" or, perhaps, what *all but a few* think right.

That response, though sensible, does not actually dispose of what seems to be bothering the critics, that is, that Version 1 contains provisions *they* think "not right". So, for example, Frailey also objects to 1.01 because "I see no reason the ETHICS would require specifications be put in writing." To this, Mechler responds as he did to the previous objection: "at present[,]

writing is the safest way of assuring transfer of data." The code's Introduction explains that the particular rules are ones that "experience has shown need express statement". Rule 1 asks software engineers "insofar as possible, [to] assure that the software on which they work is useful to the public, employer, customer, and user [and so on]." If experience has taught software engineers anything about making software useful, it should have taught them to "document" (put in writing, whether on paper or in electronic form) as much as possible, including the specifications. That (according to Mechler) is why specifications are a matter of ethics.

The disagreement between Frailey-Shaw and Mechler here may only be terminological. For Frailey-Shaw "ethics" seems to be general and uncontroversial (rather like what the ACM identified as the one-page "Code of Ethics"). The requirement to put in writing *is* something that looks too specific (and perhaps too controversial) to be "ethics" in that sense. It is more like an item in a "code of practice" (what the ACM code would put in later sections of its code—or in its Guidelines). For Mechler, however, 1.01 is not too specific or controversial to be in a code of ethics. Mechler, an engineer, had (at this time) not yet seen the ACM code. His paradigms were engineering codes. The NSPE code of ethics includes provisions that are no less specific (and, perhaps, no less controversial); that is also true of ABET's code (or, rather, of the combined Code and Guidelines). Mechler was therefore in a poor position to understand the terminology Frailey and Shaw took for granted. They, in turn, might have responded to Version 1 with more sympathy had they been familiar with the codes Mechler consulted. We have here a significant difference in "culture" (that is, in the ways of doing things taken for granted).

Mechler's response to the two critics was not totally negative, however. Shaw noted an "omission in the handling of intellectual property." But what she goes on to describe is actually much more than that:

> Piracy appears in 4.04, confidentiality and public domain IP are used inconsistently in 4.05, copyright but not patent appears as a part of a list in 4.07, trade secrets and nondisclosure are mentioned nowhere at all (thought 4.05 was probably intended to address them), there's also nothing about export restrictions, and what there is about privacy is politically loaded.

Mechler agreed to add "patent" to 4.07 (which at that time read, "Inform client or employer promptly if, in their opinion, a project is likely to fail, to prove too expensive, to violate copyright laws, or otherwise to turn out badly"). He also offered to add "export restriction" if Shaw would write the statement and specify its placement. But, "[as] for privacy being politically loaded, maybe, but from an ethical point of view we should be very aware of these conditions". He might have added that 4.07 actually says nothing about "privacy"—except in its catch-all "otherwise to turn out badly". That catch-all could equally well be interpreted to include "patent", "export restriction", and any other consideration that a client or employer should take into account before deciding whether a project should proceed.

This brings us to the last of the five objections, the one only Shaw makes, that Version 1 fails to distinguish between "absolutes" and obligations having less "intensity". Shaw seems to want a code of ethics to consist of "absolutes" (perhaps with another document providing the "aspirations"). So, for example, she says that she would "sign up to expecting an SE to refuse to

work on a practice that is blatantly unsafe or fraudulent." What she is not willing to sign to is expecting SE's to satisfy "this committee's view of good process". Mechler has the same problem with this objection that he had with the last. He cannot see what bothers Shaw: "I have never seen intensity associated Ethics: guidelines are statements to help judgment [and, for Mechler, the code is a set of guidelines]."[52] If a project is 'blatantly' something then I don't need a code: the code is for gray areas." For Shaw, it seems, codes of ethics state what everyone knows to be a "bright line". For Mechler, the code should help draw lines in gray areas. The point of a code of ethics is to raise standards, not to leave them as they are. It is (in part) supposed to state a "view of good process".

We could examine the debate between Frailey-Shaw and Mechler in considerably more detail. There is much of interest in the details. But we shall not because this exchange merely begins a debate that will continue until the code (by then, Version 5.2) is adopted three years later. We will have time enough to consider the issues first raised here—in their more mature form. We must, however, draw two (tentative) conclusions now. The first is that the debate reveals a deep division between Mechler (and, presumably, the rest of SEEPP/E) and some members of the Joint Steering Committee concerning what should (and should not) go into a code of ethics. The division does not seem to be between academics and practitioners. While Mechler *is* a practitioner, the rest of SEEPP/E (except Norman) *are* academics (more or less) like Shaw; and Frailey (while a part-time academic) had many years experience at Texas Instruments and still held relatively high rank at Raytheon. The second conclusion is that the division between Frailey-Shaw and Mechler does not seem to be one between the "low minded" and the "high-minded". The economic welfare or social status of software engineers seems to have played no part in the debate (that is, no one has appealed to such concerns to defend their position). What seems to explain who is on which side of the debate, if any thing does, is a close identification with ACM (rather than IEEE-CS), an identification itself arising from an educational background in mathematics and computer science rather than engineering. The critics (so far) are ACM's representatives on the Steering Committee. They seem to have implicitly accepted the ACM's code as the ideal by which all other codes are to be judged.[53]

## 5.8 Gotterbarn to the Rescue

On October 8, about 8:40 AM, Mechler sent Cabrera a brief email (with the amusingly errant salutation "Feline"), offering his own guess about the origin of the Frailey-Shaw criticism:

> I tried to answer the concerns as best I could. Hope it is OK. I think the change to a professional status will bring up this type of frustration from many people and maybe this individual is fighting it a little. I did not send my comments to the group; do you want me to? If there are additional comments like these[,] maybe a meeting is in order to discuss the results.

Apparently, Cabrera thought it a good idea to let SEEPP/E see the Frailey-Shaw-Mechler exchange because, at 11:33AM, Mechler emailed Cabrera again (the salutation now "Felipe"):

> I feel in tune with the group after all the reviews and comments building the product. I will send both around asking if there are additional comments or areas requiring change. Will you give me a name for the reviewers to tell the group? Task Force Review Group, etc. Second review to follow tomorrow I hope.[54]

Cabrera replied twenty minutes later:

> I am satisfied with your changes given that you are in tune with the whole group.
> I did send you a second e-mail (I believe yesterday) with more comments.
> I suggest that after you incorporate those comments, if you think it would be good, you should circulate the document by the group. These issues are tough as so many people are affected by them.

Less than two hours later, Cabrera responded again: "Task Force Review Group chaired by me is fine." So, it was Mechler who provided the name by which the critics would be known, giving them a status they would not have had as two individuals on the Joint Steering Committee and adding to the confusion some of the volunteers already had about how the Joint IEEE-CS/ACM project was organized.

The next day (October 9) Mechler sent Cabrera another brief email: "Sorry I spelled your name wrong in the first review ["Feline"]. I think the first commentator got to me a little; heard those responses too many times in my life. It seems the ethics product has developed some heat in the total effort; I got four e-mails from Don Gotterbarn but I will get to them after sending this reply." One of Gotterbarn's emails (dated October 8) was a page-long announcement (and detailed explanation) of a listserv for the "Professional Competence Standards Task Force" (PRFCMP-L-Request@UTKVM1. UTK.EDU). Everyone else in Gotterbarn's task force received one of these (or, at least, should have). Mechler responded with his new email address and (for good measure) phone number. The next email announced Version 1.0 to the working group, declaring that it was "developed by a sub group lead by Ed Mechler of the professional competence working group." The next paragraph solicited comments on the code [appended several paragraphs below]:

> We are especially interested in distinguishing between those imperatives which are aspirational and those which can be embodied in a code of practice. It would be helpful if your suggestions were accompanied by suggested rewording rather than just sending a general suggestion. Please apply your special expertise to this.

The paragraph immediately below announced there had been a suggestion "by one of their [the Joint Steering Committee's] members that they review [the code] in NOVEMBER!! so your brilliant and rapid responses are encouraged." There was a "ps" announcing Gotterbarn's belief that the "list is back in service" and everyone's name is on it. Between this letter and the code itself (unchanged from the week before) was a temporary address. From July 1996 through January 1997, Gotterbarn would be at the School of Engineering and Applied Sciences, George Washington University. He would be using a GWU email address.

Having read Gotterbarn's emails, Mechler sent off one of his own to his "Volunteers":

> Our product has been review[ed] by the Task Force Review Group chaired by Felipe Cabrera. Attached is the first review comments and my reply, again in text format. If you have any additional comments or disagreements with my comments[,] send them to me as soon as possible and I will pass them on. There appears to be a rush associated with our product so I would like additional comments ASAP. I will send the second review and comments in the next e-mail. Let me know if you get both.

He then gave his new email address (as he had with Cabrera). He sent another email to the volunteers a few minutes later, his response to Shaw attached.

Late that day, Mechler emailed Gotterbarn. One of Gotterbarn's four emails of October 9 had offered to help with the response to Frailey and Shaw. Though it was now too late for that help, Mechler thanked Gotterbarn and asked some pointed questions. Mechler wondered why Gotterbarn had asked the working group to divide Version 1.0 into "imperatives" and "aspirational provisions". Was Gotterbarn siding with the critics? Mechler also wondered whether he had missed something. Frailey and Shaw seemed to share a conception of codes of ethics differing from his own. If Gotterbarn shared that conception, it was something about which he (Mechler) should know more. Gotterbarn responded with his interpretation of events and suggested that Mechler have a look at both the ACM and IEEE codes.[55] The next morning (October 10) Mechler emailed the ACM and IEEE for copies of their codes. By 8:36 AM, he had the IEEE code. It took him another hour before he was directed to the website from which he could download the ACM code.[56]

The next day (October 11), Mechler sent Gotterbarn the same documents (his answers to Frailey and Shaw) that he had sent SEEPP/E two days before. By then, Gotterbarn also had—thanks to Frailey—received a copy of Frailey's own comments (October 4) and Shaw's (October 7).[57]

## 5.9 Farewell Cincinnatus

On October 14, Mechler received his first response from SEEPP/E to his exchange with the Task Force Review Group, a brief fax from me. It began:

> I enjoyed your exchange with Cabrera. Thought you did a good job, too, in dealing with his points. If I have anything worthwhile to add, it is that (in some cases) he didn't pay close attention to the wording, for example, 1.01 does not require more than what is "possible" and its subsidiary clauses that apply only as "appropriate".

My next (and final) paragraph asked a question: "Discussion of 4.01 says 'I agree with Dennis.' Who is Dennis? We don't seem to have his message in our e-mail file!" (Though Cabrera had tried to purge the exchange of all references to members of the Joint Steering Committee before releasing them to Mechler, he had failed. One reference to "Dennis" had survived.)

Mechler emailed me the same day (October 14). He began by sharing a joke:

> I thought you would get a kick out of the two e-mails. I do not think anyone knows that you, etc are helping. It may be unethical, but enjoyable, to answer the comments when I have two or three ethics experts helping.

Of course, neither I nor any of the other experts had helped with Mechler's responses. But some had helped with the code (and all would doubtless have helped if asked). Mechler then gave his assessment of the situation, "I now think that some in fighting occurred but do not know what. Cabrera didn't write the reviews". Mechler now thought he knew what Gotterbarn wanted. Would I have a look at the ACM code of ethics? "I'd like you to read it and comment; I have some problems with the approach." He then went on to ask why "you fax me instead of e-mail" and concluded by describing the state of his SEEPP/E files (which he had promised to the research group): "I have not transferred the e-mails to disc yet. Of course I started a new directory".[58]

On October 18, Mechler received another fax, again from me. Burnstein, Weil, and I had met earlier that day. We had "decided on one editorial suggestion in last line of Introduction" (striking "one" and replacing it with "an application that"). We hoped that would "take care of what one critic thought didn't make sense". We had also discussed "how thin email discussion of our work has been so far." We had expected more "back and forth like that between you and Cabrera". We also asked two questions. One concerned how much discussion Mechler thought had gone on "off our email site (beside my faxes)". The other revealed we had either forgotten all about Miller's email (July 16) or misunderstood what it was: "Is our working group ahead of the others? Or did we just miss their reports?" I concluded with an explanation of my mysterious preference for faxes: "It's easier for me: I know how to use fax-modem; email requires print out for secretary who then types it in again (or carrying in a disk)."

Mechler responded a few days later (October 23). He thought the proposed change in the Introduction was "OK". He would add that to the one change he had said he would make in response to "the reviewer". He was still waiting to see if there would be any other suggestions or comments. So far, there was nothing. He too had expected more debate. One of Gotterbarn's emails had mentioned a Steering Committee meeting in November, but Gotterbarn now said that it had been cancelled. "I will have a meeting with Don next month, he will be in Pittsburgh, and maybe find out what is next." Mechler thought there was a struggle over format. "Were you able to get and read ACMs [code]?.... I think Don likes that one since he worked on it." Mechler did not know how the other SEEPP working groups were doing. But he had passed on to Burnstein a copy of the report (Miller's) that Gotterbarn had sent him. He too would like to know more about the state of "the entire program". He had asked a "number of people…but no answer." Perhaps he would email them now.

I faxed my evaluation of the ACM code on October 26. I had three comments:

> First, it is, of course, a new code (1992) in an unusual format. In this respect, it resembles the 1979 IEEE code that, after much boasting and a few years of use, was junked in 1990 for a traditional format. Whether ACM's new format will prove another fad, or instead an enduring contribution to code writing, remains to be seen. On the principle of engineering conservatism alone ("avoid fads"), I think we are justified in staying clear of it for a decade or two.

> Second, the ACM's code's format could not easily be imposed on what came out of our e-mail deliberations. What we produced was, in essence, a combination of ACM's "General Moral Imperatives" (some of which, I hasten to add, seem more controversial than "moral imperatives" should be, e.g., "give proper credit for intellectual property") and the more specific imperatives in secs. 2-4. But our system of organization strikes me as more useful than ACM's. The big difference between the two codes is that the ACM's has "Guidelines" and ours does not. I think the idea of Guidelines—or, more exactly, a commentary—for the Imperatives is all right (though nothing yet suggests we need one). But the actual Guidelines do not strike me as very informative, certainly not informative enough to justify the space they take. So, I say that, inch for inch, our code [is] 3 to 5 times more useful (and user friendly) than ACM's.
>
> Third, that's my opinion based simply on looking at the documents. It is, however, not my opinion alone. The two CS faculty at IIT who have used both the old ACM code and the new in teaching both prefer the old for convenience, clarity, and simplicity. So, I think we were wise to go our own way.

This response must have chimed with Mechler's thinking, but we have no record of a response. Indeed, from here on, Mechler's records tell us very little. There is an email on November 8 from Mark Kanko, Major, US Air Force. He had taught a course in the Department of Electrical and Computer Engineering, Graduate School of Engineering, Air Force Institute of Technology, Wright-Paterson Air Force Base (just outside Dayton, Ohio). He had used Version 1.0 in class, asked his students for comments, and thought their comments might help with evaluating the code. Kanko had contacted Gotterbarn the day before and Gotterbarn had suggested he contact Mechler as well. Gotterbarn and Mechler met for the first time on November 18 at the Software Engineering Institute. They ate lunch between classes Gotterbarn was teaching (and having taped). The next day (after many days of back and forth testing protocols), Mechler seems to have received Kanko's student comments as an attachment ("As I started to read, I wondered whether you would like responses"). (We will consider the student comments in detail in 6.7.)

Mechler also wrote Gotterbarn, Laurie Werth, Kanko, and Norman ("Subject: FW: Ethics Workshop") on that day (November 19): "I had trouble through the Multiple Recipients site; host not found. I decided to form my own list with members I know of. Please one of you send this on to the rest and tell me the correct address of the list." Less than two hours later, Norman responded with some comments about the ethics workshop: "Plan on it for 1998, not 1997."

Mechler's last email from Cabrera arrived on November 25, 1996. The email consisted of the "action items" adopted during the Joint Steering Committee meeting held on Thursday, November 21. (This was the meeting Gotterbarn had thought cancelled. In fact, all that had been cancelled was the participation of ex officio members.) Mechler wrote back thanking Cabrera for the minutes, saying how sorry he was that he had not known of the meeting in advance since he would have liked to have met Cabrera (who lived in Seattle). Mechler also mentioned his meeting with Gotterbarn the previous week. Cabrera wrote back: "I am SORRY, this was supposed to go to Robert Melford. Now you ALSO know all the action items that came out of

the meeting." Almost exactly a year after he seemed, like an Old Testament prophet, to come from nowhere to save Gotterbarn's working group (and SEEPP) from failure, Mechler's participation in SEEPP's work had all but ended. We must now consider what else happened in 1996 and how that prepared the way for the very different, but equally important, achievements of 1997.

5. Appendix:

**VERSION 0**

OUTLINE:
---
INTRODUCTION
SYSTEM DEVELOPMENT
- Evaluate business affects and culture changes
- Implement Ethical evaluation.
- Assure proper Goals and Objectives
- Assure proper Development Methodology
- Assure proper Project Management
- Assure proper testing, debugging, Case Tools , GUI, etc
- Only approve safe and accurate documents
- Assure proper privacy, accuracy, property, access and people
- Assure proper estimates
- Assure specifications are fully understood
- Assure environmental issues addressed
- Promote max productivity and min costs

PROFESSIONAL PRACTICES (Himself/Herself)
- Accept responsibility in engineering decisions for safity, health and public welfare; disclose endangering/abusive factors.
- results must be judged from the standpoint of society as a whole
  - meet changes as they occur.
  - departure from the norm can be justified
  - define "usual circumstances" should be a matter of professional judgement.
  - understanding of broad fundamentals.
  - neutrality in preparation
  - not serve a special interest
  - professionalism is to put the public's interest ahead of self interest

- Put forth best effort
- Show inicitive on projects
- See through to successful finish
- Don't try to do it all yourself
- Don't ignore signs of trouble
- Don't dodge the issues
- View matters from others points of view
- Accept full responsibility
- adequate technical training and proficiency

independence in mental attitude
Due professional care
adequately planned and supervised.
Present fairly
Professional skepticism
Establishment of Quality Control policies and procedures.

Avoid conflicts of interest; disclose.
Reject bribery.
No paybacks from contracts
No political contributions, gifts, commissions, etc. for award of contract
Not accept outside work detriment to regular work
Not represent adversary interest without consent
Do not use employer equipment on outside engineering
No pirated soft/etc
Respect ownership

Service only in areas of competence
Faithful agents or trustees
Non-association with fraudulent businesses
Signature to only areas of competence
Same project-one party pay only
No decisions on services provided by self
No promotional efforts without consent
Salary appropriate to professal qualifications

Honestity in stating claims.
Objective and truthful public statements
Avoid deceptive acts
Not reveal confidential facts
Not falsify or permit misrepresentation
Admit own errors
Avoid misrepresentated statements
Responsibility to detect and report errors and irregularities

No professional compromised
Not promote own interest at expense of profession
Constructive service in civic affairs
Extend public knowledge of engineering
Prepair articles

Avoid fads
Develop ethical check lists
Develop review process

Develop contingency plans
Assure data is accurate and legitimate; kept confidentiall;
authorized use
Customer fulfillment
Recognize and report problems to proper authorities, cooperate
Inform client if project will fail

PROFESSIONAL OBLIGATIONS(Others)
Improve understanding of technology.
Improve technical competence.
Improve ethical education
Develop ethical learning
Assist collegagues in professional development.
Advance professions integrity and prestige
Support the Code
Obey the laws of the country

Not review work without other engineer knowledge
Honest criticism and credit properly.
Treat fairly all persons.
Avoid injuring others by false action.
Assure employee informed
Not attact engineer falsly

Assure only qualified signatures from others
Assignments only by educated/experience
Develop positive agreement on ownership
Assign duties to utilize for potential

Assist in development of Organization ethical environment
Assure commitment
Assure employees know
Policies and procedures
to protect passwords/files/soft/hard
to cooperate with proper authorities
to work in cometent areas
to state opinion vs fact

Assure employers and supervisors know of code of ethics
Voice concerns
Don't supplant another engr after steps have been taken
for employment
Fair and just compensation
Never invade another divisions domain without knowledge

Give fair hearings
Don't prevent better opportunity elsewhere

NOTES

[1] Interview of Mechler, June 11, 2002.

[2] Mechler filed the January 1995 CFP with his other (paper) SEEPP documents.

[3] Mechler's archive E950302A.

[4] Mechler would later join the Engineering Manager Body of Knowledge Task Force (EMBOK), though it never (as far he knows) "got off the ground". October 21, 2003 email (Subject: Chapter 4). I have found no record of a task force with that name.

[5] Interview of Mechler, June 11, 2002: "it really wasn't a planned idea…the opportunity was there and I jumped on it. I said, 'Why not?' If we build it up in such a way, and we put the things in there that we think…should be ethical issues—we might be stretching ethics—but if you say that you shouldn't accept kickbacks, what's the difference in saying, 'Well, you should follow the process to get the job done.' …most of the software people, and the people that were loading the software, building the software, testing the software, had no idea of any process. They just did it."

[6] Mechler's Archive, E950313 (Tipton); E950313A (Barbacci).

[7] Mechler's Archive, E950313A.

[8] Mechler's Archive, E950421.

[9] Mechler's Archive, E950425.

[10] Gotterbarn could not have learned of Mechler much before June 10 since a) he did not include Mechler on the list of volunteers he sent out on that date and b) he had not yet included Mechler in the profcomp list he used on that date.

[11] Mechler's Archive, E950610; E950610A.

[12] Interview of Mechler, June 11, 2002. The late date he joined (March 1995) explains both his absence from all the early lists of volunteers and his sense of SEEPP's size and organization.

[13] Mechler's Archive, E950629.

[14] Neither Mechler nor any other volunteer seems to have written a statement about what the codes said about competence. Gotterbarn had, however unintentionally, reverted to the failed strategy of "divide and conquer" (accepting the division of labor between working groups). And, as before, it failed.

[15] Mechler's Archive, E950701.

[16] Mechler's Comments on Chapter 4 (October 31, 2003). Presumably, the deadline he is referring to is the date for the November 1995 meeting of the Joint Steering Committee. But I can find no mention of that meeting date in any of the June or July memos.

[17] Mechler's Comments on Chapter 4 (October 31, 2003).

[18] Interview of Mechler, June 11, 2002.

[19] "I also understand from Jayaram that Mario thinks we are working on the project; that is why he is cc." (December 4, 1995)

[20] The email was actually sent (and received) on December 6. In those days, I wrote emails as if memos, leaving it to CSEP's secretary to turn the memo into email, often losing a day or two in consequence.

[21] Again, it seems to have been Jayaram who told Mechler that Cabrera had replaced Barbacci as chair (when Barbacci had become IEEE-CS President): "Do you have an e-mail address for Louis-Felipe Cabrera? I [Mechler] will send him of [sic] of a copy of what we have been doing and see where he wants to go. I assume you will still work with us even if our leader will not." Mechler's Archive, E960319A (email to Jayaram).

[22] The three contributors Mechler may have intended would be El-Kadi and Jayaram, each of whom had done some code summaries Mechler may have seen, and I, who had made a few suggestions. Mechler (E960708), a note to Norman a few months later, confrms the small contribution of even these three: "You have contributed more than half of the small group. Only three have contributed. My next message will point this out. Of course the small group has done at least a million times more than the large group." Mechler was politic enough not to point out in his "next message"—or in any other he sent to his volunteers—how little they were doing.

[23] See, for example, March 20, 1996 (in an email to Jayaram with cc to rest of his volunteers): "I am cc this message to the remainder of our new sub-task force SEEP[P]/E (like the name? – no imagination)….I am also sending it to the SEEPP/E because your excuses." The single P in the first use seems to be a typographical error.

[24] Mechler's reasoning turns out to have been simple: "The reason I wanted you to write the good words, Mr. Jefferson, was that I thought all the [other] participants were software engineers." Email: October 31, 2003.

[25] The earliest of these was "The Moral Authority of a Professional Code", *Authority Revisited: NOMOS XXIX* (New York University Press: New York, 1987), pp. 302-337.

[26] The most important of these are: "Thinking like an Engineer: The Place of a Code of Ethics in the Practice of a Profession", *Philosophy and Public Affairs* 20 (Spring 1991): 150-167; and Michael Davis, "Codes of Ethics, Professions, and Conflict of Interest: A Case of an Emerging Profession, Clinical Engineering", *Professional Ethics* 1 (Spring/Summer 1992): 179-195.

[27] "Civic Virtue, Corruption, and the Structure of Moral Theories", *Midwest Studies in Philosophy* 13 (1988): 352-366.

[28] Some years after this event, Gotterbarn asked me why I had not checked with him before accepting Melcher's offer. I had no clear memory but I did have the impression that Mechler at least had Gotterbarn's tacit approval. Mechler had used Gotterbarn's distribution list to send several messages, and had included Gotterbarn on some others, all without any objection from Gotterbarn.

[29] Fax (Gotterbarn to Davis), December 1, 1997 (Archive).

[30] Gotterbarn\Volunteers\Members 1995 (6/10/95) as well as Gotterbarn\SEEP VOL_10_94. Omitted from this June 1995 list are: IIT's three (Burnstein, Weil, and me), Mechler, and Sullivan. Gotterbarn seems to have begun updating the June 1995 list a few days after returning from England. See Gotterbarn\94-96 Misc, for the various working group lists including "PCOMPENT-WG list" (file date February 21, 1996).

[31] See Gotterbarn email to Davis, June 5, 2003.

[32] For those who find Gotterbarn's forgetting Jayaram's location implausible, it is worth pointing out that Mechler's April 10 email records an even more implausible oversight that Mechler himself committed (and admitted): "I should have thought of this earlier. I could have visited Burnstein [in Chicago] and Sullivan [in Washington] and I don't even travel that much." Since Mechler knew (in some sense of "knew") that Burnstein was part of IIT's research group, he should have realized that she was in Chicago just as the other two members of that research group were. But he did not realize it. Apparently, he made the same sort of mistake Gotterbarn did.

[33] How many days later? The only documentary evidence suggests that the date was at least a month later (much later than my memory suggests). On May 1, 1996, Weil wrote Mechler as if she did not know that I was to work on the code:

> The starting and stopping is exasperating but I hope you will not drop the project. At the moment you have a good list that has to be organized—that's what I meant by "categorized" weeks ago. Some injunctions have to do with competent and responsible work, "put forth best effort;" others have to do with customer, etc. A preamble about

computer engineering and developing programs is needed to give an organized list some context. I hope these comments are of use.

Mechler's response the next day (May 2) is worth quoting in full: "It is EXASPERATING but I think necessary. I will not quit until someone says so and after a fight. Thanks for the input." May 2 was the same day he wrote reminding me that I had said I "would add the 'good' words' to the ethics outline".

[34] Mechler Archive, E960325. It was Mechler who informed me of Vivian's contact (in a way that did not register at the time). See E960326, March 26, 1996: "Also Vivian has suggested a form for the ethics project and I have invited her for lunch to discuss; told her to see you. I forgot you two are at the same place."

[35] My original (handwritten) response given to the secretary for sending had the "O.K." as a question. The question mark was lost in transcription.

[36] The question of sending diskette versions arose because CSEP was having trouble receiving large attachments at this time. The cause may have been either that CSEP's electronic mailbox was too small (and some large messages just never arrived) or that the program CSEP's secretary had to use to open attachments (PINE) was too complicated for her. CSEP was then having both sorts of problem.

[37] As if to add to this confusion, the same email—"Diskette (with a somewhat revised version of code you saw is in mail)"—was sent again on July 19 (E960719) and withdrawn a minute later (E960719A).

[38] In fact, I had by now forgotten Gotterbarn's emails of the year before. I was simply following the ordinary practice of legal codes.

[39] For details, see Paula Wells, Hardy Jones, and Michael Davis, *Conflict of Interest in Engineering* (Kendall/Hunt: Dubuque, 1986).

[40] Interview of Sullivan, October 10, 2002; Interview of Jayaram, February 25, 2003.

[41] Like most of the other late additions, this was Burnstein's.

[42] Compare Mechler's comments (November 1, 2003): "shortly after the first version [of the Code] was released we received an e-mail, 'that might interest us', about one of the other subgroups [Miller's]. It was good work but we couldn't determine how to place it into our work at that time. The other subgroup was somewhat related to the original eight working groups but we thought we couldn't use their work at that time. To this day, we do not know what happened to their work."

[43] Mechler archive, E960930.

[44] October 6, 1996, p. 1.

[45] Gotterbarn\Version 0\ACODE4, p, 1.

[46] Mechler's September 11, 1996 memo treats my revisions as just another set of suggestions: "We had some last minute suggestions; I have added and attached."

[47] October 31, 2003 (Subject: Chapter 4).

[48] Gotterbarn\Version 0\ACODE4, p, 2.

[49] October 6, 1996, p.4.

[50] That is, the email to Cabrera did not go through Version 1 provision by provision giving the source for each. Mechler would provide such a provision-by-provision analysis to Gotterbarn a few months later. See Chapter 7.

[51] Shaw is a bit gentler: "The last sentence of the second-order bullets being incomplete sets of examples is pretty obscure"—but, it should be noted, not so obscure that she did not understand it. Gotterbarn's Archive, Version 0, ACODE4, p. 2.

[52] See, for example, Mechler's response to Shaw (October 7, 1996): "The question is, Where does responsibility begin? It must start at the SE Professional. I think too much 'Thou shall not' or 'Thou shall' is being read into the Code's Items. They are guide lines."

[53] This is the interpretation I now give this debate, knowing what I do of the authors of the criticism. The impression we at IIT had at the time was quite different. Weil's official report to NSF (July 21, 1998) gave this interpretation (Report on Grant #9523650):

> There followed comments from the Chair of the Task Force Review group, who criticized the draft for vagueness, for requiring the impossible or what is outside the control of the software engineer, and questioned the basis for anyone outside a company or its managers defining appropriate behavior for software engineers. The comments were presented together with replies by Mechler, who remarked on the "bleak outlook" reflected in the comments. He handled a second set of similar comments in the same way. As more comments came in critical of "soft, qualitative language" and the like, they appeared to reveal a division between the perspective of academics (members of the WG) and the perspectives of practitioners (contributors of comments).

In fact, the Task Force Review Group consisted of two members, one an academic and the other a practitioner-academic. Mechler's group may actually have had proportionally more practitioners (depending on what counts as a "unit practitioner"). Whatever the cause of the disagreement, it does not seem to be the same as reported in: Stuart Shapiro, "Degrees of

Freedom: The Interaction of Standards of Practice and Engineering Judgment", *Science, Technology, and Human Values* 22 (Summer 1997): 286-316.

[54] If "second review" means the response to Shaw, this phrasing must be a slip of memory or pen. Mechler seems to have sent out that response the day before.

[55] These emails seem not to have survived. We know about them through Mechler's email to me (October 14, 1996): "Don wanted to help me answer but one was sent and the other was being sent when I receive his e-mails….I think I finally found out what Don may have wanted."

[56] Note that this is Mechler's first contact with the ACM code even though he seems to have "asked for" that code as early as December 4, 1995. Had it become easier to get the code (or had Mechler's methods improved with Gotterbarn's advice)? It is also interesting that he now asked the IEEE-CS whether it had a code of ethics of its own and learned several days later (October 14, 1996) that it did not. Even after a year's involvement with SEEPP, he was clearly not an "insider".

[57] Gottebarn\Version 1\DFCOMMENTS (October 4, 1996); Gotterbarn\Version 1\CMTMARY—1996 (October 7, 1996).

[58] By this time, Mechler also knew "Cabrera didn't write the two reviews but two people, Dennis and Mary [did]."

# Chapter 6: The High Politics of 1996

"Politics is not an exact science."—Otto von Bismarck (1863)

"Politics is not a science…but an art."—Otto von Bismarck (1884)

## 6.1 Meeting Rogerson

While Mechler's SEEPP/E was writing Version 1 of the code of ethics, SEEPP's other activities continued. Occasionally, they intersected with SEEPP/E's; more often they did not. Their story reveals both the importance of what Mechler accomplished and some of the political problems that accomplishment bypassed or produced. SEEPP's activities in 1996 also help to explain why Melford resigned as co-chair at the end of the year, why Gotterbarn became the sole chair of a reorganized SEEPP, and why—the next year—the new SEEPP produced Version 2, 3, and 4 of the code.

On January 27, 1996, Gotterbarn received an email from Steve Barber whose help he had asked in recovering "the changes we made [in the SEEPP Task Force 'Guide to Operations'] at the 3/95 meeting in D.C." The computer that the "recording secretary" (Miller) had used to record the minutes had later failed, causing the minutes to be lost.[1] Barber had applied a UNIX "diff program" to compare the February 1994 draft of the Guide with the document Gotterbarn emailed on February 22, 1995.[2] Apparently, Gotterbarn still thought the Guide important enough to deserve this attention.

The first two days of February, 1996, Gotterbarn was in London for PASE '96. ("PASE" stands for "Professional Awareness in Software Engineering".) The conference, though a one-time affair, was a sign that understanding software engineering as a profession was not a mere American eccentricity. It was Gotterbarn's attendance at this conference that provoked Jayaram's angry email. But, more important now, it was at this conference that Gotterbarn met Simon Rogerson (who was there to give a paper).

Rogerson received a BSc (Bachelor of Science) in computer science from Scotland's University of Dundee in 1972. He worked in industry for twelve years, at various levels, on scientific and commercial management systems. He was Computer Services Manager for Thorn EMI when it announced a move to a part of England in which he did not want to live. He decided to change careers. Because the government was then (1983) trying to bring instructors with industrial experience into technical universities, Rogerson soon found a lectureship—at Leicester Polytechnic, an hour train ride north of London. The Polytechnic became De Montfort University a few years later (as part of a general enhancement of technical education in England). There Rogerson remained, adding application-oriented courses and maintaining links to industry by consulting. His industrial experience led him to see "social responsibility" as a necessary feature of doing software the right way. In 1995, he helped to found De Montfort's Centre for Computing and Social Responsibility (CCSR) and became its first director. Rogerson did not consider himself an engineer, not even a software engineer:

> In the UK, the standard job titles are “applications programmer”, “systems analyst”, and “information systems programmers”. Computer scientists are interested in the science of computing rather than the application. In the UK, software engineers are interested in building robust systems according to engineering principles. They are more technically oriented than I am. (I think the same is true in the rest of Northern Europe, though perhaps not in France.) If I had grown up in the US, I might answer differently. I’m an applications person—and, in the US, someone who works on applications is a software engineer even if he is, as I am, more interested in the way people and programs react than in the technical qualities of the applications.[3]

By 1996, Rogerson was not only CCSR’s Director; he was also (officially) a member of several SEEPP working groups, including Gotterbarn’s own. Not having heard much about SEEPP since he sent in his application on October 19, 1994, he asked Gotterbarn what was happening. From what Gotterbarn said in response, and from what Gotterbarn said in his two public talks, Rogerson concluded that “both Don and I were grappling with problems of creating the proper culture for good programming”. He therefore invited Gotterbarn to spend a semester as a guest of CCSR. Gotterbarn, then planning a sabbatical for the academic year 1996-97, accepted.[4]

## 6.2 Try, try again

A few days after PASE (February 16, 1996) and in another city (Philadelphia), SEEPP—or, to be more exact, Gotterbarn, Melford, Barber, and Miller—met from 8 AM till noon during the ACM-sponsored Computer Science Conference. All the SEEPP members present knew that SEEPP had missed its deadline. To this bad news, Gotterbarn added more. He had talked to Frailey, the Joint Steering Committee’s vice chair. According to Frailey, ACM was increasingly unhappy with the whole joint project. The other two task forces had also missed the November deadline. The whole three-footed project seemed much larger, indefinite, and difficult than ACM had anticipated. Unless the task forces made significant progress soon, ACM might desert. How the IEEE-CS would respond to ACM’s desertion was anyone’s guess, but it was certainly possible that IEEE-CS would do as ACM had. Frailey advised developing a “detailed schedule”, realistic but strict, to help get the task force working again. He wanted something he could show the ACM Council when someone complained that nothing was happening.[5] Gotterbarn also “distributed a paper [he] had written comparing the work of several professional societies.”[6]

By noon, Gotterbarn had a rough draft of a schedule. After polishing it, he sent the draft to Miller for comment (February 21). On the same day, he wrote a letter to “Messrs. Cabrera and Frailey” describing the accomplishments of the Philadelphia meeting and requesting money for two conference calls, for a meeting in the fall (at Brookings in DC), and for hiring a “single skilled writer”, or “scribe”, to assemble the documents the working groups developed into a “single document”. Miller would be the scribe: “He is a practicing software engineer who has had contacts with NASA and published widely on software testing. He is a university professor teaching computer science and working in the area of computer ethics.” Gotterbarn describes the document Miller would produce as “standards of practice, standards of conduct, and standards of

ethics". Gotterbarn also described what would happen once the document was drafted. There would be a review process involving "one or two people from each of the societies' boards". After revisions, "we would like to follow a process that was helpful in developing the ACM Code of ETHICS and Guidelines for Professional Conduct...we would like to submit the document to the membership for review and comment by publishing it in the Communications of the ACM and in IEEE Computer." Gotterbarn explained the need for a funded face-to-face meeting in the fall by noting the absence of any conference that all would normally attend and adding: "E-mail, unfortunately, is a very sterile medium for the creative exchange of ideas. (It is also very easy to procrastinate if E-mail is the only means of communication.)"[7] The Steering Committee did not approve the budget request.

On March 3, 1996, Gotterbarn emailed the revised schedule to "SEEPP work group leaders" (and to Little and me) "for comment"—using an SEI listserv (se-ethics@sei.cmu.edu). The introductory paragraph declares:

> Below is an expansion of the detailed schedule which we developed at our meeting in Philadelphia. At the meeting, I indicated that I would like to micro-manage the project for awhile, so please send me copies of the items marked with an asterisk. I am sorry for the short turn around this week, but I am leaving the country for a week on the 9th of March, and I do not want to delay things because of my travel. We have the plan and once we start to follow it, interest in and support for the project will start to increase. Best regards, don.

The first three items provide a good sample of the whole schedule:

> *March 1:
>
> Send a "hello message" to your working group; let them know that we are about to start and thank them for their continued interest in this project.
>
> *March 4-6:
>
> Develop material specifically for your work group which:
>
> 1. contains a statement of purpose for our task force (distributed to you on disk at meeting) & a straw man copy of the purpose of your working group. Invite comment from them on this.
>
> SEEPP Purpose:
> Software engineers make value-laden decisions that affect health, safety, welfare and environment. Standards and principles of ethical and professional practices will guide software engineers making these decisions
>
> SEEPP Scope:
>
> Standards, guidelines and recommended practices of ethical and professional practices for software engineering

2. Explain the difference between standards of ethics, standards of conduct, and standards of practice. Request their input about issues in your working group's domain which fit any or all of these types of standards. They need not worry about which category to put them in. We will take care of that. See 3 below.

For example:
Most professions distinguish between standards of ethics, conduct and practice. These standards include a range of statements. Standards of ethics are generally broad and aspirational and do not prescribe specific behaviors—prevent harm. Standards of conduct are more specific and standards of practice are regulative—do not release a product which has not been adequately tested. Violating this later type of standard is generally a foundation for legal action. These three types of standards exist for people as members of a profession, employees, and as individuals. We are looking for a range of statements and examples of these types of standards as they apply to professional competence. (my working group). Please send any examples of these you can think of. They can be one line statements or paragraphs. We are trying to brainstorm the standards on email. I will organize your contributions and circulate the organized list to all of you for further comment.[8]

March 8

Don will put his two cents in to coordinate all these materials that you have sent and share other's insights among work group leaders. If you do not hear from him by the 8th, he probably has nothing to say. Proceed to March 15th task.

The schedule ends thirteen items (and seven months) later with, "Oct. 15 [1996]: Latest revision from scribe to Don and Bob. Begin discussion of materials with joint committee."

This email may leave one of two impressions. The first is that it is just another of Gotterbarn's schedules, indeed, one with fewer prospects of realization than the others because the first of the deadlines, March 1 ("hello message"), was two days passed by the time the email went out. Was the schedule merely a device to buy SEEPP a little time? Time for what? The message does nothing to clear up the relation between what Mechler was doing and what the other working groups might do. Indeed, it says nothing about Mechler's group (and Mechler was not to receive a copy). Gotterbarn shows no intention of adding Mechler's SEEPP/E to the list of working groups. Yet, given what Mechler's group was doing, it did not fit neatly into Gotterbarn's Professional Competence working group (or into any other); SEEPP/E's work cut across the domains of the other working groups. The plan for eight working groups that Gotterbarn had announced as "written in jello" had (it seems) long since hardened into concrete.

The second impression this email may leave is that Gotterbarn had learned something from Mechler, though not enough. While Gotterbarn is still trying to get the working groups to develop purpose and scope statements first, he has also begun to think of the process to follow the preparation of those statements as less formal (much less formal) than he had earlier. He is now advising his group leaders to collect standards from the group, to "brainstorm", not

worrying much about what kind of standards they are coming up with or even how rough. He is also trying to set an endpoint to the seemingly interminable working-group process, a date on which everything (whatever there is) would be handed over to a "scribe".[9]

As it turned out, these details did not matter. Only Miller seems to have taken even the first step the email called for. On March 3, two days behind schedule but on the day he received Gotterbarn's email, he wrote his working group a "hello message" (one indicating, in passing, that his working group had been entirely inactive until then):

> Thank you for your interest in ACM and IEEE effort to professionalize software engineering.
>
> Ages ago (we're talking YEARS, not months) you expressed some interesting working on the Software Engineering Ethics and Professional Practices (SEEPP) Task Force, specifically the Working Group on Reliability and Safety.
>
> You may have thought this working group would never get down to work. (I certainly had MY doubts.) But it looks like we may have a plan now.
>
> Before I waste bandwidth telling you about our plans, are you still willing to work with us. I hope so. Please let me know.

On March 8, Weil emailed back: "I, Michael Davis, and Ilene Burnstein, all from IIT, are still very much aboard. Please make sure we three *each* get all the e-mail messages in the Working Group on Reliability and Safety. We will join in. I have already sent Gotterbarn a message [in response to the long email of March 3 to Working Group Leaders]. Please convey your plans to us. Best, Vivian." Miller's March 3 email even reached Mechler (who was not in Miller's working group); Jayaram (who was in that group too) had passed it on.[10]

That is the last message that IIT (or Mechler) received from Miller for several months. There is a mystery here. Gotterbarn's archive contains a March 13 email from Miller addressed to "Dear Working Group Members". March 13 is two days *ahead* of Gotterbarn's schedule for this message. Among the designated recipients (beside other working group leaders) are the members of Miller's working group (including Jayaram and "Weil, Burnstein, and CSEP [Davis]"). The email should have gone to everyone on the address list, if it was sent at all (as it seems to have been), yet no one at IIT received it. We know of it only because a copy survives in *Gotterbarn*'s files (in at least three versions).[11] Because this email was not (according to the header) distributed through Gotterbarn's list, none of the usual troubles with that list can explain what happened to it. Could it be a draft that Miller sent Gotterbarn for comment?

Gotterbarn's files contain a similar document addressed to his own working group; it seems to have been sent out two weeks late (if it was sent at all).[12] It should, like Miller's, have gone out on March 15. Mechler, a member of Gotterbarn's working group, should have been a recipient. Yet, on March 18, 1996, when Mechler emailed his volunteers the "final outline", he began: "I have not heard from Don Gotterbarn since his e-mail dated 12/15/95; if any of you have heard please let me know." Jayaram responded the same day that all he knew about "the elusive Don Gotterbarn" was his visit to Jayaram's neighborhood (February 1-2). (This was the

email Mechler passed on to SEEPP/E to explain why SEEPP/E members should visit other members when nearby.) Jayaram must also have communicated with Miller after March 3 (since the March 3 email was not addressed to Miller and yet Jayaram reports):

> There is another group chaired by Keith Miller who has contacted me re: Reliability and Safety standards-related work within the Ethics framework. You will not be surprised to note that he also cc: our beloved Don! I moaned about Don's insular habits these days to Miller in a cynical way. He seems enthusiastic with this reliability and safety work!
>
> I suggest we all produce a single e-mail message to Cabrera about Don' refusal to communicate with us.
>
> I feel that the whole exercise—plethora of workgroups is fast becoming a big joke.

On April 6 (using the ETSU list), Gotterbarn emailed his volunteers the plan that he was supposed to have sent out on March 15. Mechler (and IIT) received it, but Jayaram (and perhaps others) did not.[13] The next day Gotterbarn emailed working group leaders (and Little and me)—using the SEI list:

> Howdy,
>
> I hope you are all enjoying Spring! Cold and snow here. Please let me know how you are doing with you[r] work groups. I know the schedule we put together in February is demanding. I have already missed some of the deadlines. But it is important to get started even if we are a little late at doing it. I used to say, I will get to that when things slow down…but they never slow down. At least life is not boring.
>
> Please let me know if there is anything I can do to help with your work with the groups.
>
> Best regards,
>
> Don

Gotterbarn does not seem to have received any response to this email.

On May 15, 1996, Mechler emailed SEEPP\E: "This is the message I received [from Gotterbarn] after e-mailing Felipe Cabrera the new head of the Steer Comm. I will be sending our work soon. Please let me know if you get it." At the end of March, Mechler had decided to do as Jayaram suggested, write something to Cabrera about "Don's refusal to communicate with us". That message does not seem to have survived, nor does Cabrera's response, if there was one. But a few days later (April 8), Gotterbarn emailed Mechler:

> Ed,
>
> I am alive and apologetically slow to respond. Thank you for your work on the project. Please send your results to the list. It may serve to bring some other folks into action. I will later work your sub-groups contributions and whatever else we get into a strawman for everyone's consideration.
>
> Again, thanks for the help.

don[14]

Apparently, not much had changed since 1995. The only group, beside Gotterbarn's own (and Mechler's), to show any life in early 1996 was Miller's. But that was in March. Miller does not seem to have communicated with his working group in April, May, or June (or to have reported anything to Gotterbarn). Miller seems to have missed a number of deadlines. For example, like all other group leaders, he was supposed to send a "[revised] draft of…group's [substantive] document to members of SEEPP task force" on June 15. He did not (or, at least, there is no evidence that did).[15]

6.3 The exceptional Miller

Just because Gotterbarn sent SEEPP no emails from early April to early June, we must not suppose he was inactive. He was certainly continuing to work to make software engineering a profession. For example, he was in St. Louis, June 21-22, 1996, for the National Software Council (NSC) Forum on the Licensing of Software Professionals. Lawrence Bernstein, a member of several SEEPP working groups, a vice-president at AT&T, and also NSC president, summarized what happened by email (August 26, 1996).[16] Gotterbarn was not the only one involved in the joint IEEE-CS/ACM project present. There were at least three others (beside Bernstein himself): Zweben, by then ACM President; Chikofsky (apparently representing IEEE-CS); and Shaw (apparently, like Gotterbarn, there simply as an individual). Norm Gibbs was also there, identified as at Guilford College, North Carolina, but "recently [head] of the software education program of the SEI". Shaw argued against licensing (as she regularly did) because "software engineering is immature compared with civil or chemical engineering". For software engineering to mature, it would have to consolidate and validate its "body of knowledge". She "asked whether software engineering was closer to mathematics than engineering." She also "discussed standards" calling for "codification of our body of software engineering knowledge". Gotterbarn responded that "such a body of knowledge existed". What was lacking was an "accepted ethics of software engineering." He warned that "if the software engineering community does not step [in and] certify and license its professionals, the professional engineering community will." That would not be good because "[today's] state licensing examinations have little relevance to software engineering." Bernstein ended his report with his own recommendation for registering software engineers (rather than licensing them). Among the details of his recommendation is the requirement that registered software engineers "advocate and practice [a] code of software engineering ethics for safe systems". Except for Gotterbarn's, his is the only reference to "ethics" to appear in the report.

According to the (admittedly ambitious) schedule announced in March, Gotterbarn was to receive revised (substantive) documents from the working groups on July 15. July 15 came and went without one document arriving. Even Gotterbarn's own working group on professional competence did not report. Then, on July 16, he received an email from Miller, with a report attached, two-and-half pages single-spaced, on "Testing, Reliability, and Trustworthiness: Informed Consent and the Computing Profession". Miller had done what he was supposed to do—and he had done it almost on time. Perhaps he had even done it on time. There was some confusion. The email referred to a "grievous error" of "yesterday", misspelling the name of

Thomas Hilton, a member of the group, "whose work I found so helpful". Yet the document itself contains no references whatever—to Hilton or to anyone else. So, there seems to be no reason why the report (rather than the covering letter) should have had to be revised since "yesterday" (or why its title should be "Revision 2"). Gotterbarn does not seem to have received Miller's email of the day before, nor does anyone in IIT's research group. Perhaps, the retracted email had never been sent (or, perhaps, it went where so much of 1996's email already had).

Miller's document ("Revision 2, July 16, 1996, 11am") seems to be what Gotterbarn would describe as a standard of practice. There are eight unnumbered sections, beginning with an Introduction of four paragraphs describing a problem of "computer professionals"—determining "how good is good enough"—and a proposal to solve the problem through a procedure of "informed consent". The next section, three sentences long, states the "Guiding Principle":

> A software developer should obtain informed consent from the client. A software user should have available information about the safety and reliability of software. The public should be able to obtain detailed information about the testing and reliability measures of any commercial software.

The section after that ("Definitions") defines five key terms in the Guiding Principle ("software", "software developer", "client", "user", and "public") and sends the reader to the next (five-paragraph) section, "Informed Consent", for the definition of that term. The section defining informed consent is followed by "Implementation of Informed Consent" (a single six-line paragraph) and a (half-page long) section "Contents of an Informed Consent" (divided into three subsections: "I. Product Measurements", "II. Process Description", and "III. Assessment Implementation". The final two sections (each a paragraph long) are "Contract Ramifications" and "Informed Consent Documents as Public Archives". The last quarter page of the document (three short paragraphs) is a commentary entitled "Why I Think This Might Work".

Miller's document is clear, compact, and (seemingly) practical. It shows that Gotterbarn was justified in suggesting him as SEEPP's "scribe". The document is, however, probably not the work of a "working group". Miller speaks as "I", not "we"; and the very quality of the exposition suggests that he has not had to make the compromises that working with a committee generally requires. He mentions no votes or approval (though he sent a copy to every member of his group, as well as to Melford, Gotterbarn, and himself). He seems to have solved the problem of getting his working group to work by doing the work himself (perhaps consulting with Hilton or just reading something he wrote). Miller's solution to the problem of getting the working groups to work, the only one achieved so far, is further evidence that division into working *groups* may have been a mistake (even apart from the assurance problem discussed in 4.8).

Miller's report also suggests a problem with developing standards of practice within a project to make software engineering a profession. The document never speaks of "software engineers" but of "computer professionals" and "software developers". The definition of "software developer" is quite broad: "any professional involved in the production of computer software; this includes but is not limited to designers, coders, testers, managers, and salespeople." The definition of "software" is even broader. Software includes "computer programs, internal documentation, design documents, test plans, user manuals, online help, and the like." "Professional", while not defined, seems to mean nothing more than someone hired to

do a certain job, “a role” having “responsibilities and privileges”. Put all this together and even a technical writer editing a user manual may count as a “software developer” for the purposes of Miller’s document. Though the very broadness of definition is part of what would make the document useful, the document is not, as written, a contribution to the professionalization of software engineering. (Technical writers, for example, belong to a different profession—with its own code of ethics.) Indeed, Miller’s document says nothing about software engineering as such. That, perhaps, is why we hear nothing more of Miller’s achievement.

## 6.4 Getting back into the loop

Gotterbarn does not seem to have received or sent any messages to SEEPP, to his own working group, or to Mechler’s group during the rest of July, all of August, and (almost all) of September. This is odd because the schedule he sent out in March had the following deadlines:

> Aug 15: first draft of complete, coherent document to task force.
> Sept 1: revisions from task force to scribe.
> Sept 15: Scribe submits revisions to working groups and task force
> Oct 1: suggestions back to scribe from working groups and task force.
> Oct 15: Latest revision from scribe to Don and Bob.
> Begin discussion of materials with joint committee.

As September came to an end, Gotterbarn (now on sabbatical leave at George Washington University) would have had reason to think that (once again) he would have very little to show the Steering Committee when it next met. Miller’s document, whatever its merits, was not the document promised, “a code of ethics and professional practices”. It was, at best, a modest element of a much larger code of professional practice. The other seven official working groups, including Gotterbarn’s own, had reported nothing. Mechler’s group had last reported to Gotterbarn in April. What that group had shown him then (Version 0) was not a “coherent document” but a disordered list, neither a code of ethics nor a code of practice. Since then, Mechler seemed to have disappeared. The project seemed doomed.

Then, on September 30, Gotterbarn received Mechler’s email announcing “the [attached] final product of the Ethics group”. Since Cabrera received the same email (as did Mechler’s working group), SEEPP was, in a way, now fifteen days ahead of schedule: “discussion of materials with joint committee” had (it seemed) begun. Cabrera soon sent the code to the rest of the Steering Committee. During the next week, there was that flurry of emails described in 5.6-5.7: Frailey’s critique (October 4), Mechler’s response (October 6), Shaw’s critique (October 7), Mechler’s response to Shaw (October 7), and some lesser exchanges between Mechler and Cabrera. Because Frailey had sent a copy of his critique to Gotterbarn when he emailed it to the rest of the Steering Committee, Gotterbarn had seen it on October 4, even before Mechler had. In a few days Gotterbarn had gone from worrying that there would be no code of ethics to worrying about whether the code would be rejected—without ever passing through relief at having something to show. Frailey had seemed to be a strong supporter of SEEPP’s work. If he was as

unhappy with the code as he sounded, others on the Steering Committee must (it seemed) be unhappier. Something had to be done, and soon, or the code would be dead.

Evidence of that wider unhappiness arrived on October 7, Shaw's critique, which Frailey had himself passed on to Gotterbarn with the covering note: "Don, I think the steering committee would like to hear some response to my comments and Mary's from those who formulated the draft code of ethics."[17] Frailey was right. Indeed, what was needed was a careful response like the paper Gotterbarn had helped with Anderson and Johnson prepare for the ACM Council when the ACM code was in danger. That would take time to prepare, especially if the response was to come from SEEPP (as it should). Right now, Gotterbarn seemed to be in an odd position. He was neither on the Steering Committee's mailing list for discussion of the code nor on SEEPP/E's. He was "out of the loop". He knew what passed between those two groups if, but only if, some member—Frailey (for the Steering Committee) or Mechler (for SEEPP/E)—sent him a copy of the relevant document. He seemed to have lost control of the process for which he (as chair of the Professional Competence working group) or he and Melford (as SEEPP co-chairs) were responsible.

The next morning (October 8), Gotterbarn announced to his working group a listserv he had been preparing for some time, PRFCMP-L-Request@UTKVM1.UTK.EDU.[18] More than a single-spaced page of instructions explained how to use it. Three hours later, Gotterbarn used the listserv to send Version 1 to his working group and begin a process of review that might at least convince the Steering Committee not to make up its mind right away:

> Dear professional competence working group member,
>
> Below is a draft of a code of ethics developed by a sub group lead [sic] by Ed Mechler of the professional competence working group.
>
> We are soliciting comments from you on the code. We are especially interested in distinguishing between those imperatives which are aspirational and those which can be embodied in a code of practice. It would be helpful if your suggestions were accompanied by suggested rewording rather than just sending a general suggestions [sic]. Please apply your special expertises to this.
>
> A copy of this draft has been seen by the steering [committee] and it was suggested by one of their members that they review it in NOVEMBER!!, so your brilliant and rapid responses are encouraged.
>
> Thanks for your help and continued interest.
>
> don Gotterbarn

This message, though evidently written in haste, is more complicated than it may seem on first reading. Gotterbarn' explicit reference to distinguishing between imperatives that are "aspirational" and those that can be "embodied in a code of practice" suggests that he thought that the chief problem with Mechler's code was that it had failed to distinguish between those provisions which, being merely aspirational, belonged in a code of ethics and those which, being meant to be enforced, belonged in a code of practice. Frailey and Shaw both seemed to object to

“aspirational” provisions because they thought they were intended to be enforced. They both seemed to envision a code much like the current ACM code. Gotterbarn too had expected SEEPP’s work to end in something like that document. It was, after all, “state of the art” in code writing. The problem, then, was how to move Version 1 in that direction. Technically, SEEPP was the body that should do that. But working through SEEPP would have meant working with Melford, SEEPP’s co-chair, and working with Melford would be slow. Melford was now taking a very long time to respond to messages, when he responded at all. Indeed, Gotterbarn had not heard anything from Melford since they had met in London in February. Gotterbarn certainly did not have the time to wait for Melford now. Gotterbarn did not need Frailey to suggest that the Steering Committee might vote on the code in November. Such a vote would only be natural given that they had the code in hand and the other two task forces were still far from having anything like it to present.

Gotterbarn nonetheless seems to have done his best to make clear that he was only addressing his working group, even using the salutation “Dear professional competence working group member” rather than his usual “Volunteers”. This satisfied the letter of the requirement that he address SEEPP only with his co-chair’s consent, but escaped the consequences. His working group included most of the members of the other working groups, including all the chairs of SEEPP’s working groups. He could reach almost everyone, certainly everyone who mattered, while officially addressing only his own working group.

For the same reason, Gotterbarn explicitly treated what Mechler liked to think of as a working group, as a “sub group”. Gotterbarn would have had to consult Melford to deal with someone else’s working group, even a spontaneous and unauthorized one like Mechler’s. But, as a “sub group”, Mechler’s group was Gotterbarn’s business alone. There was no need to involve Melford—and therefore no need to wait until Melford responded.

## 6.5 Politicking

After this email, Gotterbarn wrote another, this one to Mechler personally. In it, Gotterbarn offered to help with drafting a response to Frailey and Shaw. Mechler did not receive that response until the next morning. By then, he had already responded. An important opportunity to mollify the critics had been missed. Gotterbarn nonetheless tried to explain to Mechler the politics of the Steering Committee, urged him to take a look at the two codes the Steering Committee would likely use as standards against which to test Mechler’s, and otherwise tried to prepare him for the work ahead.

Mechler did not get around to sending Gotterbarn a copy of his comments until two days later (October 11). By then, Gotterbarn had received both from Frailey. When Gotterbarn read them, he had been appalled. They did not strike him as polite, cool, or carefully argued. They were not at all likely to soften the opposition. They also seemed explicitly to deny the distinction between an (aspirational, general) code of ethics and (mandatory, specific) code of practice that Gotterbarn, like Frailey and Shaw, took for granted. For Mechler, a code of ethics was mandatory—and as specific as it could be. Did Mechler’s working group think the same way? Or was Mechler (more or less) on his own? Gotterbarn could not tell. Over the next few days, indeed, over the next few weeks, no one in the Professional Competence working group responded to his October 8 call for criticism. Did that mean that everyone in the working group

(including Mechler's "sub group") approved of the code? That no one cared one way or the other? Or merely that no one in the working group had time to read and comment on so long a document? Or that the message had vanished into the ether?

Last, Gotterbarn decided to see whether he could prevent the Steering Committee taking up the code in November. He emailed Frailey that he had "just got your response to the code of ethics" and agreed "with many of the concerns expressed. The original plan was to distinguish codes of ethics from codes of conduct and codes of practice."[19] Gotterbarn then explained that the code "was developed by a sub-group led by Ed Mechler", that he (Gotterbarn) was "surprised when it was distributed to the steering committee before going through the kinds of revision and distribution we [had] with the ACM code", and that he was therefore "concerned about your suggestion to approve by November". While "sending it directly to the top is one way to get things moving"), there are "sensitive issues that need to be worked out and some restructuring of the code that needs to be done before it is ready":

> For example, given the current ambiguous state of copyright legislation, that item has to be filled out in the code. Are we ultimately to have three types of code—ethics, conduct, and practice or follow the ACM model of a code and put standards of conduct in the guidelines. For a code to be effective, there needs to be sanctions such as removal from membership in a society. Which society is sponsoring this and will there be similar sanctions from each society. Before the ACM code was approved it was submitted to the membership in the CACM [*Communications of the ACM*] and responses were solicited.

Having thus reminded Frailey (the ACM-appointed vice chair of the Steering Committee) that "that is not how we do things at the ACM", Gotterbarn admitted, "I am belaboring my point—one month from draft to approval seems rather fast." He then added one other consideration: He could not be at the November meeting (because he will "be attending an international computer ethics conference").[20]

This email seems to have had its intended effect. Within a few minutes of receiving it, Frailey forwarded Gotterbarn's email to Cabrara (with a cc to Gotterbarn and everyone else on the Steering Committee). The covering note: "as you can see from the attached, the proposed code of ethics we received had not gone through the ethics task force and thus can best be considered a draft. I suggest the steering committee be apprised of this and that we send it to Gotterbarn with an indication that it is in much too immature a state to approve at this point."[21] The immediate crisis had passed.

On October 18, Frailey emailed Gotterbarn again. He needed "a few things [by October 25]". One was a description of contacts with "external" organizations (apparently, meaning "foreign" rather than non-ACM organizations). Frailey helpfully provided a "model entry" using PASE'96 (Professional Awareness for Software Engineers, a conference held in 1996). Frailey needed this list because he was putting together "a chronology of all external organizations with whom the software engineering activity has interacted" for Chuck House, the new ACM president. The other item Frailey needed was an "update on plans for 1996", noting that the plan Gotterbarn had drawn up in January called for "a meeting in Washington in October or November to have the group do a final run through of the document" and "[delivery of] the document to the steering committee."[22]

Gotterbarn's response tells us at least as much about his state of mind at this time as about the state of SEEPP. After acknowledging that Frailey would not "be happy with the level of detail that follows", he explained why he could not then do better:

> I am now in Leesburg VA teaching at George Washington University [Ashburn campus] while on Sabbatical leave….This is my first sabbatical and it is an organizational nightmare. Some of the names and email address[es] you requested are locked in a file cabinet in Tennessee but they are also in one box or another of materials I have been forwarding to England [where the second half of his sabbatical year would be].

Gotterbarn's list of "external contacts" is short, consisting in large part of the panelists with whom he shared the stage at PASE'96 (beginning with the two, Tracy Hall and Colin Myers, whom Frailey had mentioned in his "model entry"). There are, however, four exceptions. One is Rogerson. Gotterbarn will be at his Centre "developing standards of practice in project management and… revising and developing materials for the task force." The second contact is an unnamed representative of the (British) Institute of Electrical Engineers (IEE). Gotterbarn was "looking for guidance on the development of standards and soliciting their participation in the development and the review process. The individual I spoke with declined the offer and indicated that such participation was not a high priority for IEE." The third contact is "Tony Cowling, one of the developers of the BCS [British Computer Society] undergraduate software engineering degree; the fourth, "Duncan [Langford], an author of an applied computer ethics text." These last two are described as among other "volunteers in England who have already been contributing to the documents developed."[23] Gotterbarn does not say what the "documents" in question are or how Cowling and Langford contributed. (The two had, of course, nothing to do with the drafting of Version 1, though Langford would soon begin contributing.)[24]

Gotterbarn concludes this email with another note (marked by another salutation):

> This does not fit easily into your template. First week of November I am going to a computer ethics conference and will do some specific recruiting, in particular I want to involve people from the US based center for Computing and Social Values [the Research Center for Computing and Society at Southern Connecticut State University]. Later in November I am doing a videotape for the SEI on ethical issues in project management and will take the opportunity to meet with some of my working group who are centered in Pittsburgh [e.g., Mechler]. <- I know Pittsburgh is not a foreign country.[25]

The information Gotterbarn provided Frailey, however thin, clearly assumes that SEEPP's work will not end in November. Gotterbarn will be working on the code while in England, starting January 1997. But the information, whatever its assumptions, sounds strangely like what Gotterbarn could have written a year or two earlier. He cannot report that his working group is actually working on the code. Or even that his working group (or any other) is working. The most he can report is that he will recruit people (for SEEPP's working groups)—as if he had not recruited enough people already.

6.6 More comments on Version 1

Between the time Frailey asked for a report and the time Gotterbarn answered, Gotterbarn received two responses to his request for comment on Version 1. One, from Langford, an early SEEPP volunteer, began by declaring the code "impressive". The eight specific suggestions for improvement that followed the declaration show how impressive Langford actually thought the code. He had one overall suggestion: "I'm not sure that 'assure' is the right word; would 'ensure' be better?" The specific suggestions, though all minor, reveal a careful reading of the code. He proposed amending Rule 1.03 ("Asssure that they are qualified, by education and experience, for any project on which they work") by appending "or propose to work"; Rule 1.14 ("Avoid fads, departing from standard practices only when justified") by inserting "technically" before "justified"; Rule 2.01 ("Disclose to appropriate persons any danger…") by inserting "actual or potential" before "danger"; and so on, always adding to the precision of the rule or raising the standard it set.[26]

The same day (October 22), Gotterbarn also received a long email from Milton Fulghum, Senior Staff Engineer at FlightSafety International (Missouri), another early SEEPP volunteer.[27] Fulghum offered no overall evaluation of the code, but his page and a half of suggestions, as specific as Langford's, clearly take the code's overall structure for granted. Most of his suggestions are organizational—and minor—for example: Clauses 1.05 and 1.14 should be combined since "1.14 seems to be a specific case of 1.05." Clause 2.03 "might better fit under another rule, perhaps Rule 3". Clause 2.04 should be moved to Rule 1. But one of the organizational changes is major. He notes that many of the clauses in Rule 6 (in particular 6.07, 6.08, 6.09, and 6.14)

> address management issues instead of peer-to-peer relationships as implied in 'Colleagues' title. The clauses [in 6] should focus on peer-to-peer issues. Perhaps another rule should be drafted to address management issues.

Fulghum ends by wondering how "does this [code] compare with" the new ACM and IEEE codes.[28]

On November 7, Mark Kanko, Major, US Air Force, used the listserv to ask whether "you are still accepting inputs".[29] Kanko had assigned the code to his "graduate students (some with real-world experience, some not)." He would be happy to send the results. Gotterbarn responded two days later (using the listserv): "I would love to see them. Sorry about the delay in responding, I was at Ethicomp 96 all week [Madrid]. I would also appreciate your comments."[30]

The next day (November 10), Gotterbarn received another response to his call for comments (an email sent directly to his "gwu" address). It came from one "Peter Ron Prinzivalli, IEEE member", of San Jose, California. Prinzivalli was the Principal Software Consultant of his own company. He had been involved in "software development, QA, maintenance, and production" since 1966, served as Treasurer of the IEEE-CS Standards Activity Board 1984-87, and been a member of Gotterbarn's working group (and three others) since November 17, 1994.[31] Like Kanko, he was an engineer rather than a computer scientist.[32] And also like Kanko, he had never before responded to any of Gotterbarn's emails. But his response now, over two pages of "comments" and "edits", followed by some extremely kind words, made up for his two years of silence.

Prinzivalli's concerns seemed different from the other commentators', positive as well as negative. He began by asking what would happen next "after this working group is finished with it". Will it go to IEEE-CS "Governing Board and DC Staff"? What will be done about its "propagation and maintenance"? He then wonders to whom it will apply: "programmers with no formal education or programmers who are not software engineers by university/trade school degree?" What he really wants to know is whether "software engineer herein mean[s] EE-SWEng"? As an engineer, he is concerned that the term "engineer" be reserved for engineers who have graduated from an ABET-accredited program—or, at least, something very close, not for mere "programmers with no formal education". (He is therefore interested in how the code's "term"—"software engineer"—connects with the body of knowledge and curriculum the Steering Committee is also working on.) The specific changes he suggests are (mostly) quite minor and distinctly his own. For example, he suggests substituting "must" for "shall" in the Introduction ("Software engineers shall commit themselves…"); "authorities" for "persons" in 2.01; and "required" for "appropriate" in 2.02. But some of his suggestions resemble those Frailey and Shaw made. For example, Prinzivalli prefers "documented" for "put in writing" in 1.01 (to "cover all electronic documentation"); he does not think Rule 1 is "possible" (unless there is an "acknowledged 'professional' process of supervision"); and he is not sure what 6.11 means ("Not supplant another software engineer after steps have been taken for employment"). Overall, Prinzivalli's suggestions seemed to move the code language away from the ethical and toward the legal, but his overall evaluation was much the same as Langford's and Fulgum's:

> Don, congratulations on getting the process this far. I have participated in several IEEE software development standards, Software Standards Conferences, as well as on the Society's Standards Activity Board. Consensus and participation is always difficult to shepherd into productive useful Standards and Codes. Good luck with the final stages of authorization and publication.
>
> I'd suggest that the Society provide sessions at all of our Conferences and Tutorials to propagate the Code. Each member and non-IEEE member should be able to attend these sessions, and the Society should maintain a Registry of all individuals who have form[al]ly agreed to follow the code.

6.7 Kanko's student comments

Gotterbarn met with Mechler for lunch, November 18, a get-acquainted meeting without agenda or definite outcome. When Gotterbarn got back to Leesburg, Kanko's "comments", all thirty-two single-spaced pages of them, were waiting. Kanko had asked his class of fifteen students to read Version 1 and a) comment on the code generally, b) identify "no more than three individual imperatives that should not be part of the code, and c) identify "no more than six of the individual imperatives that are aspirational". Kanko's students were all officers enrolled in a course in the Department of Electrical and Computer Engineering, Graduate School of Engineering, Air Force Institute of Technology. They were therefore likely all to have trained as engineers and to have had most of their practical experience, if any, inside the Air Force. Their responses were unlikely to provide a good indication of how software engineers generally would

respond. Worse, Kanko's assignment itself also seemed unlikely to provide a representative evaluation of the code. The assignment included some clear biases, for example, that there must be at least three imperatives that should not be part of the code. At best, Kanko had provided one data set among the many that would be needed to evaluate the code—but, on that November day in 1996, Kanko's was by far the largest data set Gotterbarn had. Gotterbarn read on.

The responses, each about two-pages long, did not sound particularly "military". Though some of the students explicitly identified themselves as young officers ("lieutenants"), they do not seem much more obedient to orders than their civilian counterparts. One student flatly refused to identify any imperative to be removed: "I feel all the imperatives have a place in the code." Another identified only two imperatives for removal. Another, after listing three imperatives for removal, suggested adding one ("1.15 Assure the quality of the product meets the quality requirements of the user and/or employer"). And one went far over the quota (naming five imperatives for removal plus "most of 3.0x"). Kanko's students responded with equal unpredictability to part c of his assignment (six aspirations).

Overall the students' evaluation of Version 1 was mostly either favorable (7) or "mixed" (6), with only one student clearly negative and one that simply did not state a position. The overall pattern of recommended deletions is interesting. None of the imperatives suggested for removal are in Rule 2 or Rule 7. The most targeted (with 6) is 1.14 ("Avoid fads, departing from standard practices only when justified"); the next (with 5) is 5.07 ("Only accept a salary appropriate to professional qualifications"). There were two imperatives three students would delete: 4.01 ("Provide service only in areas of competence") and 5.11 ("Serve in civic affairs constructively"). Five imperatives were named twice (1.13, 4.02, 5.03, 5.10, and 6.11). Thirteen others were named only once (among them some, but only some, that Frailey or Shaw complained about). Ignoring the answer "most of 3.0x" (that is, one student's vague suggestion to remove most of the provisions under "Judgment"), most of the code (49 out of 71 imperatives) seemed to have passed muster—or, at least, to have passed without comment.

The students' attempt to identify "aspirational" provisions suggests some trouble with the term or the concept. Some students explicitly defined "aspirational": a provision was an aspiration if it stated a standard the satisfaction of which is "hard to measure" (because it is open to interpretation, is not a "yes/no", or not easy for others to check performance on). For others, the aspirational imperatives seem to be those that, though "laudable", are "unrealistic" or at least hard to satisfy. One student identified the *Rules* as aspirational (even though he also thought there was one too many). One said that the entire code was aspirational; one singled out "everything in 7"; and one just did not answer. Not one of the students equated "aspirational" with "general" (as Gotterbarn had in his March 3 email).

Most of the specific imperatives identified as aspirational were either under Rule 1 Product (30) or Rule 5 Profession (22). The imperatives under Rule 6 Colleagues came in a distant third with only five mentions. The imperative most often identified as aspirational was 5.01 ("Associate only with reputable businesses") with six mentions. Runners up were imperative 5.04 ("Help develop an organizational environment favorable to acting ethically") and 1.04 ("Assure proper goals and objectives on any project on which they work") both with five. Two more, 1.02 ("Assure that they understand fully the specifications for software on which they work") and 3.01 ("Maintain professional skepticism with respect to any software or related documents they are asked to evaluate"), received four mentions. Four others received

three (1.05, 1.06, 5.03, and 5.11); seven, two mentions; and the remainder one or none (ignoring the answer "all the rest").

Kanko's data suggest at least four comments. First, his students were not inclined to delete imperatives they regarded as aspirational. So, for example, the imperative most often picked for deletion (1.14) received only two votes for "aspirational", and the highest ranking imperative for aspiration (5.01) received no votes for deletion. Kanko's students seem not to object to a code of ethics containing aspirations.

Second, while we might explain the high number of mentions of imperatives under Rule 1 by location (first come, first picked), we cannot explain the status of imperatives under Rule 5 that way. Something else made the imperatives under Rule 5 salient—both for deletion and as aspirations. We can only guess what. The students do explain why they chose the imperatives they chose but not why they did not choose the imperatives they did not choose.

Third, the students seem *not* to have found the distinction between "standards of practice" and "aspirations" a natural way to divide up the code's imperatives. They also seemed not to share an unequivocal understanding of what the distinction is.

Fourth, the students seem not to have paid much attention to the clause introducing the imperatives. Their evidence that a particular imperative is aspiration almost always comes from the language of the imperative itself. So, for example, though the imperatives under Rule 1 are introduced with what might be taken as aspirational language ("In particular, software engineers shall, as *appropriate*:"), no student quoted that language as evidence for the imperative being aspirational (not specific, requiring interpretation, or the like). In this respect, and only in this, Kanko's students seem much like Frailey and Shaw. They too seemed to read the code one "bullet" at a time rather than as an integrated document. The students did not even try to read the particular clause in light of the general rule it came under. Apparently, Version 1 is not as easy to use as Mechler (and, apparently, the rest of SEEPP/E) thought.[33]

## 6.8 Computer scientists v. engineers?

Gotterbarn now had a riddle to solve. Mechler seemed to have a point when he had responded to Frailey and Shaw by talking about engineering codes of ethics. All the engineers responding to the code at all (and even most of the engineering students) liked the code. Whatever the merits of the individual criticisms Frailey and Shaw made, there seemed to be little, if any, overlap between them and the "engineering response". For engineers, the code had no obvious flaw. Gotterbarn could see why computer scientists like Frailey and Shaw would respond differently. The codes they were used to didn't look much like Version 1. But it was not possible to explain their response simply by the contrast between engineers and computer scientists. Langford, a Research Fellow in the Computer Laboratory of the University of Kent at Canterbury, held a Ph.D. in Computer Science (University of Kent, 1991)—with an undergraduate degree in social work.[34] In no sense an engineer, he too liked the code. Could the Frailey-Shaw response constitute nothing more than a statistical anomaly?

The Steering Committee (minus all but one ex officio) met in Pittsburgh on Thursday, November 21. The "action items" (November 25, 1996) suggest that the Committee had been thinking about some of the things Mechler had said in defense of Version 1, especially his references to the "PE code of ethics". The first action item (for Cabrera "due 11/24/96") was to

send “Stuart Feldman [a computer scientists at IBM] a copy of the “code of Ethics of Professional Engineers and Civil [Engineers]”. Melford (in town for another reason) undertook to rewrite (by “1/20/97”) “the proposal for a Code of Ethics and Professional Practices for Software Engineers following the canonical form of those for Professional and Civil Engineers.” He was to “produce a document that complements and expands those previously mentioned [to] align this proposal with that of sister disciplines [and be] different only when there is need to be different.” Chikofsky (by “1/1/97”) was to [identify] and secure relevant statements that address the issue of licensing software engineers [and send] the material to Felipe and e-mail to the whole steering committee regarding the findings.” Melford also had two large assignments concerned with *licensing*. By “11/24/96”, Melford was to provide Cabrera with “an initial statement of the question whether software engineers should be licensed engineers, to be used as the basis to present the forthcoming discussion on this subject.”[35] By “2/20/97”, Melford was also to report “on the Computer Engineer example [including] in the report their top-level definition of the field.” (Mixed in with these items were similar items, but far fewer, concerned with the body of knowledge. For example, Frailey and Shaw were (by “12/13/96”) to produce “a set of paragraphs that describe the field of real-time software engineering”.

Though addressed to “Ed”, this email was intended for Melford (as Cabrera explained in a later email sent the same day). It is worth quoting in full:

> Many thanks for your participation in our meeting last Thursday. I appreciate your involvement and your work.
>
> I did regret to be unable to talk longer with you about the stressing life situations that you have been going through. I look forward to spending some time at the next meeting in February.
>
> FYI I enclose the complete list of action items that you helped us draft. There are several that belong to you and [I] will appreciate it very much if you let me know if clouds appear in the horizon and the work cannot be done.

Melford must have read this email soon after his return to California, looked at the demands his consulting business had been putting on him for well over a year (“stressing life situations”), and realized that (given that business was unlikely to slow down any time soon) he could not do what he had promised. Within a few days, he informed Cabrera of all “the clouds…in the horizon” and resigned as SEEPP’s co-chair. On December 10, 1996, Gotterbarn learned that he was now the sole chair.[36]

Frailey seems to have been the one to inform Gotterbarn of his new authority. This he did by phone on December 12.[37] Frailey also asked Gotterbarn to prepare the three documents that Melford had undertaken to provide. Gotterbarn did not accept the assignment as Frailey originally presented it. Other phone calls followed. It was only on December 15 that they agreed on what Gotterbarn was to do. There would be three documents: 1) a statement describing the roles, structures, and functions of codes of professional ethics; 2) a detailed comparison of Version 1 to other relevant professional codes (including but not limited to engineering’s); and 3) a revised code. Gotterbarn would have nothing to do with licensing—even though he had long been an advocate of licensing. The code was in enough trouble without connecting it with the extremely contentious question of licensing.

## 6.9 Sole chair

On December 18, 1996, Gotterbarn began to exercise his new powers. He sent an email ("Dear Task force members") using both a formal list "seep workgroup" and individual addresses for himself and most of the original members of SEEPP (Little, MacFarland, Miller, Sullivan, Barber, and Weisband—with Melford omitted, of course). The email also listed Mechler and his seven as individual addressees, with Cabrera and Frailey each receiving a cc. (Noticeably absent is Gotterbarn's own Professional Competence working group.) Though Gotterbarn alone signed the email, there is no other indication of Melford's departure. (Gotterbarn believes he notified the task force leaders of it two days before this email, in an email that has not survived.)[38]

The list of addressees is nonetheless a bit odd. Laurie Werth, one of the original eight SEEPP members is missing. One of those on the list, Weisband, had not attended a SEEPP meeting, or even responded to an email, in well over a year. Little, McFarland, and Sullivan had also long since ceased to attend SEEPP meetings. Apart from Gotterbarn and Melford, SEEPP had become only Steve Barber and Keith Miller. Having heard nothing from Barber in months, Gotterbarn had emailed him, asking whether he received Version 1 ("I am really looking forward to you[r] perspective on it"). Gotterbarn also asked for a report "telling me the status of your group…and where you are in developing a preliminary statement." Barber wrote back on October 21: "We should probably talk. My life is getting busier, and SEEPP is having a hard time finding it to the top of the list. My strategy of procrastination is not serving any of us particularly well." Having left law for software a year before, he found giving time to SEEPP increasingly hard.[39] So, by November, he too had become one of SEEPP's phantom members.

Gotterbarn's December 18 email begins on a high note ("It is time to move the code to the next step") but soon gets down to business ("It is evident from the comments [presumably, those of Frailey and Shaw] that we need to do some foundation building"). He then describes in detail the three documents he had promised to deliver by February 1997 and sets out another one of his schedules ("The plan"):

> By the first week in January
>
> Send references to sources for imperatives in draft code to Don.
>
> (Don) write draft statement about professional codes
>
> By Mid January
>
> Comment on draft about codes
>
> (Don) distribute statement about relationship of SE code to other codes
>
> By beginning of February
>
> Achieve consensus on structure of code and start to revise it in the light of that consensus.
>
> By Mid February
>
> Revised draft of code ready for comment by Steering Committee.
>
> By the middle of March
>
> Have a discussion draft ready for distribution on the net and in professional journals

Have survey form ready for distribution with the code.

Gotterbarn concludes the email: "I know this [is] a lot to ask during this time of the year, but we have at last achieved something positive with the draft code and if we lose this momentum I am concerned that we will lose all credibility. But even more importantly, if we don't get the job done the concept of professional ethical standards will also lose credibility. I look forward to getting your responses…[etc.]"

Mechler responded the next day (December 19, 1996), giving Gotterbarn his new email address (again), and providing seven sources for the "Code we developed". The first on the list was the "IEEE and IEEE CS Code of Ethics"; the next (identified as the source of the "initial outline") was the National Society of Professional Engineers Code of Ethics (presumably, for version 0). The ACM code was listed fourth, just before the American Institute of Certified Public Accounts Professional Standards Manual. Listed last (after the "PA State Registration Board of Professional Engineers, Land Surveyors and Geologists The Code of Ethics and the Law" was a publication of the American Society of Mechanical Engineers, *The Unwritten Laws of Engineering* by W. J. King. The next email Mechler received from Gotterbarn was a two-page form letter (January 1, 1997): "You have been added to the PRFCMP-L List".[40] The only evidence that Gotterbarn ever received Mechler's December 19 email, if he did, is that Gotterbarn later used the references in the memo on sources of (7.3). But, by then, Mechler had resent the list (January 30, 1997). Mechler's December 19 email seems to be another of the lost emails of 1996.

No one but Mechler seems to have responded to Gotterbarn's December 18 email. So, as Gotterbarn prepared for the second semester of his sabbatical, the prospects for SEEPP would have looked neither as bright as two years ago nor as dark as one year ago. On the dark side, SEEPP no longer seemed a large organization—or, at least, one that responded enthusiastically to Gotterbarn's call to action. On whom could he count? Mechler, Miller…who else? Gotterbarn would be spending the remainder of his sabbatical, the first half of 1997, in England—at Rogerson's center. Staying in England meant he could attend few of the meetings he usually attended, could not count on many of the face-to-face meetings that seemed to have kept SEEPP going so far, and so might have the least help when he needed help the most. What could England offer instead? On the bright side, Gotterbarn at last had a draft of the code of ethics, a much simplified decision process, and a relatively undemanding schedule during which to work on the code. He could organize SEEPP as he thought best, without having to wait for Melford's approval. He was now in charge.

6. **APPENDIX:** CODE OF ETHICS FOR SOFTWARE ENGINEERS v1.0 September 1996

INTRODUCTION

Computers now have a central and growing role in commerce, industry, government, medicine, entertainment, and ordinary life. Because the utility of computers depends in large part on the instructions written for them, those who design, develop, and test software have enormous opportunities both to do good and to cause harm. To assure, as much as possible, that this power will be used for good, software engineers commit themselves to making the design, development, and testing of software a distinct, beneficial, and respected profession. In accordance with that commitment, software engineers shall adhere to the following standards of conduct. The seven main paragraphs state general rules. Each subsidiary clause is a specific application of its general rule, one experience has shown needs express statement; but no set of subsidiary clauses exhausts the general rule.

Rule 1: PRODUCT. Software engineers shall, insofar as possible, assure that the software on which they work is useful to public, employer, customer, and user, completed on time and at reasonable cost, and free of significant error. In particular, software engineers shall, as appropriate:

1.01. Assure that specifications for software on which they work have been put in writing, satisfy the user's requirements, and have the customer's approval.
1.02. Assure that they understand fully the specifications for software on which they work.
1.03. Assure that they are qualified, by education and experience, for any project on which they work.
1.04. Assure proper goals and objectives for any project on which they work.
1.05. Assure proper development methodology on any project on which they work.
1.06. Assure proper management on any project on which they work, including proper procedures for control of quality and risk.
1.07. Assure proper estimates of cost, schedule, personnel, and outcome on any project on which they work.
1.08. Assure adequate documentation on any project on which they work, including a log of problems discovered and solutions adopted.
1.09. Assure proper testing, debugging, and review of software and related documents on which they work.
1.10. Assure that software and related documents on which they work respect the privacy of those who will be subject to the software.
1.11 Assure that raw information used in software is accurate, derives from a legitimate source,

and is used only in ways properly authorized.

1.12. Assure ethical, economic, cultural, legal, and environmental issues are properly identified, defined, and addressed.

1.13. Promote maximum productivity and minimum cost to employer, customer, user, and public.

1.14. Avoid fads, departing from standard practices only when justified.

Rule 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 any danger that the software or related documents on which they work may pose to the user, a third party, or the environment.

2.02. Approve software only if they have a well-documented belief that it is safe, meets specifications, and has passed all appropriate tests.

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

2.04. Cooperate in efforts to correct problems in software or related documents.

2.05. Be fair and truthful in all public statements concerning software or related documents.

2.06. Not put self-interest, the interest of an employer, or the interest of a client or customer ahead of the public's interest.

2.07. Accept full responsibility for their own work.

Rule 3: JUDGMENT. Software engineers shall, insofar as possible, 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 skepticism with respect to any software or related documents they are asked to evaluate.

3.02. Reject bribery.

3.03. 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.

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

3.05. Neither solicit nor accept a contract from a governmental body on which a principal or officer of their employer serves as a member.

3.06. Participate in no decision of a governmental or professional body, as a member or advisor, concerned

with software, or related documents, in which they, their employer, or their client have a financial interest.

3.07. Disclose to all concerned parties those conflicts of interest that cannot reasonably be avoided or escaped.

Rule 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 competence.

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

4.03. Use the property of a client or employer only in ways properly authorized.

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

4.05. Keep as confidential information gained in their professional work that is not properly in the public domain.

4.06. Identify, document, properly report to employer or client any problem in the software or related documents on which they work.

4.07. Inform client or employer promptly if, in their opinion, a project is likely to fail, to prove too expensive, to violate copyright laws, or otherwise to turn out badly.

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 consent.

Rule 5: PROFESSION. Software engineers shall, in all professional matters, advance both the integrity and reputation of their profession. In particular, software engineers shall, insofar as possible:

5.01. Associate only with reputable businesses.

5.02. Assure that clients, employers, and supervisors know of this code of ethics.

5.03. Support software engineers who do as this code requires.

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

5.05. Report violations of this code to appropriate authorities.

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

5.07. Only accept a salary appropriate to professional qualifications.

5.08. 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.
5.09. Not promote their own interest at the expense of the profession.
5.10. Obey all laws governing their work, insofar as such obedience is consistent with the public health, safety, and welfare.
5.11. Serve in civic affairs constructively.
5.12. Improve public knowledge of software engineering.
5.13. Share useful software-related knowledge, inventions, or discoveries with the profession by reading papers at professional meetings, by publishing articles in the technical press, and by serving on the profession's standard-setting bodies.
5.14. Make no political contribution, gift, or commission for award of a contract.

Rule 6: COLLEAGUES. Software engineers shall treat all those with whom they work fairly. In particular, software engineers shall, as appropriate:

6.01. Assist co-workers in professional development.
6.02. Review the work of other professionals only with their knowledge.
6.03. Credit fully the work of others.
6.04. Criticize the work of others in an objective, candid, and properly-documented way.
6.05. Give a fair hearing to the opinion, concern, or complaint of a colleague.
6.06. Assure that employees are informed of standards before being held to them.
6.07. Assure co-workers know the employer's policies and procedures for protecting passwords, files, and other confidential information.
6.08. Assign work only upon considerations of professional qualifications.
6.09. Provide for due process in hearing charges of violation of an employer's policy or of this code.
6.10. Develop a fair agreement concerning ownership of any invention an employee makes.
6.11. Not supplant another software engineer after steps have been taken for employment.
6.12. Attract employees only by full and accurate description of the conditions of employment.
6.13. Offer only fair and just compensation.
6.14. Not prevent a subordinate from taking a better job for which the subordinate is qualified.

Rule 7: SELF. Software engineers shall, throughout their career, try to enhance their own ability to practice their profession as

it should be practiced. In particular, software engineers shall continually endeavor to:

7.01. Improve their knowledge of recent developments in the design, development, and testing of software and related documents.
7.02. Improve their ability to create safe, reliable, and useful software at reasonable cost and within a reasonable time.
7.03. Improve their ability to write accurate, informative, and literate documents in support of software on which they work.
7.04. Improve their understanding of the software and related documents on which they work and of the environment in which they will be used.
7.05. Improve their knowledge of the law governing the software and related documents on which they work.
7.06. Improve their knowledge of this code, its interpretation, and its application to their work.

NOTES

[1] Interview of Barber, November 14, 2002, answer to question 8.

[2] The difficulties in the way of reconstructing what had happened in March 1995 were considerable; their description reveals as much about the importance Gotterbarn still assigned the Guide as about Barber:

> Let me know if you need help interpreting the diff output, but hey—man diff. The diff output shows more changed text than there actually is because I re-justified the paragraphs I changed. I've annotated the diff output to indicate the Guide paragraph in which the change belongs.
>
> These changes have been reconstructed using my notes made at the time. The notes were not complete, and in many cases were but a few words intended to convey the idea or the change. I've written new language using the notes as a basis, but do not pretend that my language here replicates the language we came up with together at that meeting.
>
> There were two instances where I merely noted the word "change" with an arrow pointing to the affected text. I have not attempted a reconstruction in these instances, but have merely highlighted the area. Perhaps the identification will jog someone's memory; I have no independent recollection of those changes.

Gotterbarn\SeepOld 94-96\draft of guide Jan 96.GDE (January 27, 1996).

[3] Interview of Rogerson, February 24, 2003, answers to questions 1-3. Rogerson statement that he is not an engineer is odder in England than in the US (and therefore suggests that he has given the question considerable thought). The British Computer Society is a member of the (British) Engineering Council. Any graduate of an accredited computer science program is as eligible to become a Chartered Engineer as the graduate of a mechanical, civil, or electrical engineering program. See Trevor Burridge, "Certification in the United Kingdom", *IEEE Software* November/December 1999: 46.

[4] Interview of Rogerson, February 24, 2003, answer to questions 5-6.

[5] Gotterbarn, General Comments on Chapters 2 and 3, June 6, 2003.

[6] Gotterbarn\SEEPP1996-97\SEEPBIL\Meet1 ("Dear Messrs. Cabrera and Frailey").

[7] Gotterbarn\SEEPP1996-97\SEEPBIL\Meet1.

[8] While three-part distinctions between kinds of codes are common, this particular one (between ethics, conduct, and practice) is not. Compare Mark S. Frankel, "Professional Codes: Why, How, and with What Impact?", *Journal of Business Ethics* 8 (February-March 1989): 109-

115; or the various articles in Margaret Coady and Sidney Block, editors, *Codes of Ethics in the Professions* (Melbourne University Press: Carlton South, Victoria, 1991), important works Gotterbarn might have drawn on.

[9] Of course, this "scribe" is a standard feature of IEEE standards development. SEEPP's Guide 4.1.4 specifically provides: "The WG Chairperson should then identify an individual to author an initial draft. The author should be permitted to prepare the draft with or without additional input or assistance from any other WG members, at his or her discretion." Gotterbarn Archive, 94-96 MISC, OPGUIDE feb 96. The only novelty is having a *task-force* scribe, another indication that the working groups were not working as they would in any ordinary IEEE standard-writing project.

[10] Mechler's archive E960318B.

[11] Gotterbarn\SEEP1994-96\Miller to R&S WG; and Gotterbarn\SEEP\1996-97\Miller (with the time stamp and other formatting of an emailed actually sent); and Gotterbarn\SEEP1996-97\COMPET from GD and KM.

[12] Gotterbarn\SEEP1994-96\MCOMPACS (in four different forms, but all with files dated April 6, 1996).

[13] Mechler (March 20, 1996): "It is funny that you did not get Dons e-mail, I thought it was caused by a[n] e-mail from you and you were on the original mailing list." Jayaram was on the original mailing list, but his address, like almost everyone else's, had changed a bit since. Gotterbarn could only update the list if informed of a new address.

[14] IIT Archive, April 8, 1996.

[15] Of course, missing such deadlines may, or may not, mean a group is working. By June 15, Mechler was in the last stages of polishing Version 1. Though he had sent a copy of most drafts of Version 1 to Cabrera when distributing them to SEEPP/E, he had not communicated with Gotterbarn for several months. The same may, as we shall see, have been true of Miller.

[16] Bernstein seems to have used a mailing list having nothing to do with SEEPP. At IIT, only Burnstein received Bernstein's message.

[17] Gotterbarn\Version 0\ACODE4.

[18] There were a number of reasons for this new listserv, including the advance of technology. But one of them was that Gotterbarn, on leave, was no longer in a position to keep the close watch on what came in and should go out as he had been when back at East Tennessee. As he later recalled (email, February 17, 2004): "I went on leave—Summer & Fall 1996 I was at George Washington University—it took two months before they had me on E-mail so I was using a commercial E-mail address for a while—need to notify the Univ of Tennessee [in order

to do anything with listserv].” Gotterbarn does not recall when he received his official GWU email address. The earliest email from Gotterba@seasva.gwu.edu seems to be September 9, 1996 (to Mechler). E9696.

[19] This, of course, is Gotterbarn’s reconstruction of events. In fact, the original plan does not seem to have distinguished between codes of ethics, conduct, and practice—until Gotterbarn’s meeting with Barbacci and Melford in January 29, 1994 (3.4).

[20] Gotterbarn\Version 1\ACODE1.

[21] Gotterbarn\Version 1\ACODE1.

[22] Gotterbarn\SEEP1996-97\RENEPTW.

[23] Gotterbarn\Steering Committtee\df-myplan.

[24] Cowling is neither on any of Gotterbarn’s early address lists nor on the final list of code “authors”. There is nothing in the archives to suggest he ever contributed anything. The first of Langford’s contribution seems to have arrived a day or two before Gotterbarn reported him as a contributor (as explained below).

[25] Gotterbarn\Steering Committtee\df-myplan.

[26] Gotterbarn\Version1\CMTDUNCAN.

[27] Gotterbarn\1994-96 MISC\SEEPVOLS-list 2-95, pp. 26-27 (Fulghum).

[28] Gotterbarn\Version1\CMTFULG.

[29] This is the Kanko (“the Major”) who had a year earlier told Mechler (Ch. 4.6) that he was not in the Professional Competence working group but in Sullivan’s on Privacy. Apparently, neither Kanko nor Mechler ever informed Gotterbarn of his preference for another working group. Now, the failure to cleanse the mailing list of the silent paid off. The moral here seems to be: *you never know what people will do if you give them the opportunity to do something both useful and within the bounds of what they want to do*.

[30] “ETHICOMP” is the acronym for the “International Conference on the Social and Ethical Impacts of Information and Communication Technologies”, a conference held annually in a European city. The first, organized by Rogerson and Terry Bynum (Southern Connecticut University) was held in 1995 at DeMontfort.

[31] Gotterbarn\1994-96 MISC\SEEPVOLS-list 2-95, pp. 56-58 (Prinzivalli).

[32] Gotterbarn\SEEP 1994-96\VOLAFIT 2-23-95.

[33] I too was disappointed. The lesson I took from the response to Kanko's assignment is that codes are not self-interpreting; those who are to use a code should be taught how to interpret it. Gotterbarn may have taken a similar lesson from the comments, adding to his reasons for wanting the Preamble to provide a detailed guide to interpreting the Code.

[34] Gotterbarn\1994-96 MISC\SEEPVOLS-list 2-95, pp. 42-43 (Langford); Email (Langford to Davis), March 15, 2004.

[35] Apparently, licensing was a subject that the Steering Committee could not separate from its thinking about professionalization—even though the subject had officially disappeared (two years before) between the Board of Governor's approval of the "blue ribbon committee" (2.4) and that committee's first report (2.5).

[36] Gotterbarn thinks my treatment of Melford here—and in the preceding chapters—too lenient (Gotterbarn\2003 SEEPP\Three more Chapters 5-7, September 1, 2003):

> I guess what irritates is that your description of his "resignation" seems to indicate that he was working hard all along and that it just became too much work for him. Even when his company was floundering- he had the same 'work' pattern. Remember this had been smoldering since February of 1996 at the ACM meeting. Did he ever even organize his own working group? etc. Two documents of boiler plate at the front end of the project.
> No need to dig deeper, just please change the tone of the poor down trodden over worked Melford description - Without further corroboration you can see that Melford set himself a bunch of tasks on the spur of the moment at that meeting, but fortunately this time he realized that he was- (MD pick one) totally out of the loop as far as the code was concerned and as far as any work with the other working groups was concerned, unprepared to do what he said he would do, realized his own work pattern related to this project- set dreamy goals without intention to do what needed to be done to deliver, said things as usual just to impress the IEEE power brokers, and THIS TIME did so in an individual way where HE would be responsible...

Gotterbarn's interpretation may be right (based, as it is, on closer contact with events), but I find my interpretation better given the evidence I have—and the feeling, hard to defend, that I am reconstructing events akin to those that led to a messy divorce, the sad end being read back into all that went before.

[37] Gotterbarn\History of SE Code\History expanded. See also Gotterbarn Chapter6cmt: "I remember the following. Before I left Virginia, in the basement of the house we were staying in, via email and phones, there was a discussion about the future of SEEPP. I was given charge of SEEPP and the tasking for the three papers before I left."

[38] IIT Archive, December 18, 1996. In any case, Gotterbarn notified everyone in SEEPP (and its working groups) of his assuming SEEPP's chair (and of his English address) in an email of January 6, 1997.

[39] Gotterbarn\Steering Committee\Barber. See also Interview of Barber, November 14, 2002: "I stopped paying much attention to SEEPP sometime in 1996. I switched back to software from law in September 1995. The career change left me with less time for SEEPP and other 'extra-curricular' activities — I had to focus on learning a lot of software technology that I missed during my five-year excursion into the law."

[40] The (automatic) listserv may have been working on its own. On January 6, 1997, I received this message from Laurie Werth (through the listserv): "I would like to be added to the mail if I am not already on it."

# Chapter 7: Winter Whirlwind, Version 2

I bear orders from the Captain:
"Get you ready quick and soon,
"We must all be together
"At the rising of the moon."
—Irish folk song

7.1 Statement on codes of ethics

On January 3, 1997, Donald Gotterbarn and his wife Shirlee arrived at London's Heathrow Airport. Not sure what to expect of English weather from January to July, they had packed one of everything, filling four large suitcases and two carry-on bags. They took a bus to St. Pancras Station and there boarded the Northern Express, struggling with their luggage so much that the conductor called ahead to Leicester to be sure a luggage handler would be ready to help them off the train. An hour later, they were stepping onto the platform at Leicester, a city of a quarter million people, once a center for manufacture of knitwear, hosiery, and shoes, now the home of many small high-tech companies and two young universities. The station is a short walk from the nineteenth-century City Centre and from De Montfort University's Leceister Campus. Simon Rogerson was waiting on the platform (along with the promised luggage handler). Having greeted them, Rogerson led the way to his car, loaded as much of their luggage as he could while leaving room for them, and put the remainder in a taxi. The taxi following, he drove to the terrace house, a half mile from campus, where they were to live for the next six months. A light snow was falling. The city looked like a postcard.

On Monday (January 6), Gotterbarn arrived at the suite assigned the Centre for Computing and Social Responsibility (CCSR) on the tenth floor of the James Went 2. (Went was a split-level high-rise. One side of the split was Went 2; the other side, a half-storey down, was Went 1.) A gray, rundown, and undistinguished building, marked for demolition early in the next decade, Went contained both offices and classrooms—as well as a snack shop and other amenities. CCSR's suite consisted of one large open square with desks for a Research Associate (Ben Fairweather), three students, and the "webmaster" (a technician who maintained the server for CCSR's website). Windows along one side made the open square bright. Having shown Gotterbarn around, Rogerson led him to the second floor of Went 1 where his office, formerly a storage area, was waiting. A (more or less) triangular room with a desk "3-4 feet long", a computer, a book case, and two chairs, it was just large enough for two people of modest proportions to meet. But it was not without advantages. It had a picture window the length of its longest wall, was convenient to toilets and elevators, and had one other advantage, though Gotterbarn would not appreciate that one for several weeks: As far as DMU was concerned, the office was still a storage closet. When Rogerson left his faculty office just down the hall to meet with Gotterbarn, he (in effect) disappeared from DMU. No one would look for him in a storage closet. Indeed, no one but an initiate would look for Gotterbarn there either. The two of them could work in Gotterbarn's office even for long periods without interruption.[1]

The preliminaries over, Gotterbarn was soon checking his mail on the DMU (desk-top) computer.[2] A message he had been waiting for—from Cabrera and Frailey— was there. Addressed to "Software Engineering Profession Steering Committee, Task Force Chairs, and Ex-Officio Members of the Steering Committee", it formally announced what Gotterbarn had agreed to two weeks before:

> In order to expedite the remaining activities assigned to this task force, we have asked Donald Gotterbarn to assume full chairmanship. The task force has the following activities to complete by February 14:
>
> - complete a draft code of ethics and professional conduct that has its roots in the codes currently in use by other engineering professions as well as the codes of the IEEE and the ACM
> - document the architecture of this code so that reviewers can understand the components, which parts are derived directly from other codes, and which parts are specific to software engineering
> - document the rationale for the new parts of the code (those not derived from existing codes or which depart from existing codes in substantive ways)
>
> Other activities for the remainder of the year will be outlined in a plan that Don will distribute next week.
>
> We understand that for a variety of good reasons, many members of this task force have been unable to participate at the level originally planned. We specifically authorize Don to drive the task force to completion, making whatever changes in membership or structures are deemed necessary to accomplish this task.

It was now official. Gotterbarn was no longer tied to an IEEE co-chair or to the IEEE standard-writing procedures. He was free to act. If SEEPP failed now, he alone would be responsible.

Gotterbarn forwarded the Steering Committee's email to his working group's listserv, adding his own covering message. The message's first paragraph announced "some changes in the SEEPP task force" and directed readers to the forwarded message from Cabrera and Frailey. The second paragraph (in part) explained the new listserv:

> In order to address these tasks [the three listed in the Cabrera-Frailey message] I have formed a combined working group [sic] collected under a single email list called prfcmp-1. This explains why several of you have received email saying you were added to this list.

Gotterbarn then promised to send "a copy of the plan" the next day and "hope[d]" to send "a draft statement of the roles and functions of professional codes of ethics…by Thursday [January 9]." Last, he informed the reorganized "working group" that he was "now working at" CCSR, that he had a new email address (dgot@DMU.AC.UK), and that the listserv should not be used for a message directed to an individual (since many more than that individual would see it).

Gotterbarn did not make much of it at the time and many, perhaps most, of those active in SEEPP's work may not have noticed (or cared), but Gotterbarn had just reorganized SEEPP. Everyone who was not already in Gotterbarn's working group was now in it. Though Gotterbarn said nothing about the other working groups, a careful reader might have noticed that the name given the new listserv implied that Gotterbarn's working group had taken over: SEEPP (the task force) had become the "*Professional Competence Standards* Task Force". The members of the other working groups had all been incorporated into one entity which (apparently) Gotterbarn now indifferently called "working group" or "task force". Gotterbarn had abandoned "divide and conquer". SEEPP now had the unitary organization of Mechler's SEEPP/E. (Gotternbarn would not again mention the other working groups, except once—in 8.1—to declare them dead. )[3] He had, however, not yet found a replacement for SEEPP itself—(at its best) a small group of advisors with whom he could meet face to face several times a year. Within a few days, he would.

The next day, Gotterbarn distributed Version 1 through the listserv. The covering note began "Dear task force member" (as would all subsequent messages he sent through the listserv). The note asked for help with "[annotating] the code with references to similar or identical imperatives in other codes"; it promised "a draft of the codes of ethics statement" the next day (Wednesday). There was no mention of "the plan" promised the day before (and, in fact, Gotterbarn had decided not to distribute it until the Steering Committee approved).[4] Gotterbarn gave no indication of when volunteers should turn in their annotations or how the work was to be divided among them. (They would, of course, know that they had to get their work in before *Gotterbarn's* deadline, but not the last date before that on which what they did would still be useful.) Perhaps Gotterbarn expected to set deadlines and divide the work once task force members volunteered. If that was what he expected, he was to be both disappointed and surprised.

On January 8 (as promised), Gotterbarn distributed the draft "code of ethics statement". Its covering note said it was "a brief document describing the roles and functions of codes of ethics in an emerging profession" that he wanted "to modify…so that it can provide some guidelines for us as we revise the draft code of ethics and determine how next to proceed". Gotterbarn concluded by inviting "comments on its content and style". Again he gave no deadline for response. The statement, about three single-spaced pages, seems intended as a summary of everything then known about the functions of codes of ethics (or, at least, of codes of ethics in "emerging professions"). But the eight references at the end suggest something much narrower. Three of the eight, including two to Gotterbarn's own previous work, are explicitly about codes of ethics in software engineering.[5] Three more are computing-related—Anderson's paper on the ACM code, Jacques Berleur's on the code of ethics of the International Federation of Information Processing (IFIP), and Johnson's *Computer Ethics*. The last two references are to textbooks in *engineering* ethics—Johnson's anthology (1990) and *Ethics in Engineering* by Mike Martin and Roland Schinzinger (1989).[6] These two represent the best texts in engineering ethics of the 1980s, but neither they nor the rest of the references reveal a wide familiarity with the literature of professional ethics or even an updating of Gotterbarn's earlier reading in engineering ethics.

Gotterbarn was no doubt doing the best he could in the few days the Steering Committee's schedule gave him. The Committee had, in fact, allowed much less time than a

mere counting of days on the calendar would suggest. During his first few weeks in England, some time had to go to setting up house, learning his way around campus, and identifying resources he would need to draw on latter. His sabbatical leave had two large projects in addition to completing revision of the code of ethics. He was to "work up the concept of Software Development Impact Statements" and develop a method for "ethical\professional risk analysis". He was also making one round trip a week to the United States to teach a ten-week course at the National Security Agency (in a Washington, DC suburb), giving lectures in various classes at DMU, and preparing papers for conferences. Gotterbarn even lost a day or two fighting a virus that infected his computer.[7] Like all those in SEEPP (and its working groups), Gotterbarn was always an unpaid volunteer, his work on the code having to compete with activities that paid the bills. That was true even during a sabbatical year.

Gotterbarn's statement began by noting a "negative standpoint" on the function of codes of professional ethics as well as a positive. The negative understands codes as "merely a self serving attempt to generate a positive public image" or "to establish a moral minimum" rather than a "complete" guide to professional conduct. While granting that these "negative judgments" must be kept in mind when preparing a code, the writing of a code should (according to Gotterbarn) focus on the positive functions. The positive functions can be achieved using one or more of three sorts of standard: 1) those that apply to everyone in virtue of a common humanity (honesty, for example); 2) those that apply by virtue of one's special role (or skill) as a professional (for example, the obligation to serve society); and 3) those that apply because of the special structure of a particular profession. Gotterbarn offered no example of standards of the third sort (in this statement or in the earlier work he referenced) but he seemed to regard such standards as (more or less) deducible from the particular profession's powers, skills, and role (just as service to society is deducible from the special role of professions as such). Gotterbarn also did not indicate whether these three sorts of standard correspond to what he elsewhere distinguished as: code of ethics; code of conduct; and code of practice (for example, in his email of October 8, 1996, to Frailey).[8] Practical (rather than scholarly) considerations seem to control what he says here, the most important of which would be satisfying the Steering Committee.

The core of Gotterbarn's statement is a list (and explanation) of the (positive) functions of codes under six headings: "INSPIRATION", "GUIDANCE", "EDUCATION", "SUPPORT FOR POSITIVE ACTION", "DETERRENCE AND DISCIPLINE", and "ENHANCE PROFESSION'S PUBLIC IMAGE". The statement seems designed to lay a foundation not only for defending the final code (whatever its form) but for rewriting Version 1 to make it more like the ACM's code (which explicitly sorted standards according to something like Gotterbarn's three-part plan—with enforcement adding a forth part ). The statement also seems (if less obviously) a justification for not connecting the code with licensing. The statement's concluding paragraphs explicitly limits the software engineering code of ethics to the first four functions. The code would *not* be designed for deterrence or discipline (as a code designed for licensing *would* be) because "[at] this stage…, the disciplinary function is being taken over by the law." The code would also not be designed to improve the profession's image because "[in] most contemporary codes, the attempt to keep a perfect public image at the expense of quality development has been abandoned." A code designed neither for disciplining errant professionals nor for improving the profession's public image is unlikely to have much use in licensing.

By 1997, the most important text in engineering ethics included one function not on Gotterbarn's list, one especially important to an "emerging profession" and the very one Mechler had stressed in his exchange with Frailey and Shaw: setting a higher standard than law, market, and morality would otherwise require.[9] The higher standard (a requirement, not an "inspiration") should if followed, should improve the profession's image (that is, give it a reputation for doing work of a quality higher than law, market, and morality ask of others). Once a profession had such a reputation, government might well want the service the profession provided to be provided only by members of the profession (as a way of assuring consumers or public a certain level of quality). Might we then wonder why Mechler did not suggest that Gotterbarn add that function of professional codes?[10] And, since the textbook in question lists me as authority for that function of codes, we might also wonder why I also said nothing at the time. Understanding why neither Mechler nor I said anything at the time may help us understand why no one else did either.

One of my reasons for not responding is embarrassingly pedestrian: I did not immediately receive the email. In those days, I only received email at the office. Just after New Years day, I had outpatient surgery on the big toe of my right foot (to fuse two joints). Though not major surgery, it did mean I had to use crutches for six weeks and could not drive. So, that January, I worked at home until the end of winter break (the Tuesday after Martin Luther King Day). Only then did I take a cab to my office, meet my class, and check my mail. By that date (January 21), Gotterbarn's statement was almost two weeks old, stale mail that had to compete for my attention first with more recent mail and then with all the demands of a new semester.[11] Without a deadline, Gotterbarn's request for comment had a low priority. Of course, I did put his email in my (paper) file, intending to respond. Within a few days, however, that intention drowned in the new wave of emails Gotterbarn was already preparing. Late January was a bad time to get much of a response from me—and perhaps from other academics.

That was, however, not the only reason I did not respond. I was back in my fly-on-the-wall mode, intrigued by what Gotterbarn was trying to do and wanting to see what would happen. I wanted there to be a code of ethics for software engineers. Without the code, there would not be much interest in the process of writing the code (or, rather, trying to write it). While I had a fatherly interest in Version 1, it was the interest a father takes in an adult child who should be able to prosper on her own. Whatever Gotterbarn's errors, they were, it seemed to me, not his alone. They were the common wisdom of several decades, and would no doubt continue to influence thought about codes for several decades more. There was then no pressing need for me to say again what I had already said. There should be other opportunities; writing the code had been a slow process. There was, it seemed to me, no reason to expect that to change any time soon. Had Gotterbarn personally sought my advice (as Mechler had), I probably would have responded quickly, but I had no reason to respond quickly to the general call for comment. There did not seem to be any deadline (except the date binding Gotterbarn, February 14, that, residing in an earlier email, I did not recall); and there were others, more directly involved, to respond—Mechler, for example.[12]

## 7.2 Mechler helps

Why didn't Mechler respond? Perhaps he did not receive Gotterbarn's January 8 email. He never mentioned it in later emails and there is no copy in his files. Or, perhaps, he forgot it as soon as he saw what it was. His interest was the code, not theoretical discussions of codes. What we do know is that Mechler took seriously Gotterbarn's request of January 7 for help annotating the code (though he did not tell Gotterbarn). On January 28, 1997, with the deadline for submission of everything to the Steering Committee approaching (February 14), he emailed Norman:

> The request from Don, dated 1/7/97, Draft Code, asked for annotation. I am doing a number of other codes but have trouble with the one you sent AICPA Standards Manual because it was extracts. Can you annotate the Code from the manual? Let me know.

Two hours later, Norman responded: "I don't have the manual any more. I guess I could get it from the University Library again, but I might not be able to get it done in time." Early the next morning (January 29, 1997), Mechler told Norman to go to the library only if he wanted to—and then, after explaining further why he needed the "exact reference", complained that the annotation "is taking forever", asked "Have you heard anything from the list since Don sent his last e-mail?", and concluded by setting the "end of the week" as his deadline for "sending out the references". By 11:09 AM (after several more messages back and forth), Norman announced, "Good news. I have the original that I sent you and it has references to sections and paragraph numbers from the AICPA Manual….Apparently, I'm more organized than I thought!" Mechler then suggested that Norman "annotate Dons e-mail of 1/7/97 and put it to the list [that is, send it out through PREFCMP-L]. I will do the same with mine." Late that afternoon (after an exchange of emails in which each admitted to being "an anarchist", that is, a person not much given to keeping orderly files), Mechler returned to Norman's question concerning a deadline: "the only dead line I have is Feb 14th. Did you get the additional schedule mentioned in the forward e-mail from Don?"

The next morning (January 30, 1997), Mechler sent his annotations to the listserv (PREFCMP-L). He cited five "source codes" (those belonging to the IEEE, NSPE, Project Management Institute, ACM, and Institute for Certification of Computer Professionals). He assigned at least one (and up to all five sources) to each of the particular rules. For example, under 1.01, he listed "PMI IIb— ACM 3.4". Mechler made no attempt to provide a source for the Introduction or for any of the seven general Rules. He prefaced the annotation with a caveat. A reference may mean that the particular provision of the code is either:

> Exactly as stated in another code
> Paraphrased from other codes
> Expanded for SE from other codes
> Idea generators from other codes.

So, if readers "decide another reference within the codes annotated is better" or "have additional codes not mentioned", they should add the reference. But, anyone who disagrees with a reference should email Mechler and "we will discuss it".

Mindful of Murphy's best-known "law" ("If anything can go wrong, it will"), Mechler waited a half hour and then emailed Gotterbarn to see whether the annotations had arrived. Mechler also took the opportunity that this email offered to ask two questions he had wanted to ask ever since he received Gotterbarn's January 7 email:

> 1. What was the intent of the statement: "we understand that for a variety of reasons, many members of this task force have been unable to participate at the level originally planned." In Felipe and Dennis e-mail you forward[ed] to us?
> 2. What is the plan for the rest of the year they requested in the same e-mail. I do not know if you sent anything yet.

Mechler also emailed Norman to see whether he had received the annotated code. About noon, Norman responded that he had received it. Sure that all was well, Mechler went on to other things. But the next morning (Friday, January 31), he received a message from Weil at IIT: "SOS. The Code of Ethics annotated toward other codes was not attached. Instead came the words 'Unable to print this part'." Mechler investigated and responded about twenty minutes later:

> It is something to do with the list. I am on the list twice using CompuServe and Exchange and both came with warning about MIME. But I received both. Here is another copy. Let me know if you received it.

Having sent this message, he wrote PRFCMP-L: "There appears to be some problem with the SEEPP List sending my annotated copy of the Code. If you didn't get it, please let me know at emechler@ero.eqt.com."

Seven minutes later (about twenty-four hours after Mechler first sent his annotations), Gotterbarn emailed the list a message with a table attached (comdes.doc). According to the message, the table not only incorporated Mechler's annotations but extended them to seven other codes. A half hour later, Mechler responded: "Can not open comdes.doc. Please resend or send as a text file." Mechler also emailed Norman directly asking whether *he* was able to open Gotterbarn's file. Norman took a half hour to reply: "My VMS mail cannot read MIME either. However, I can convert it to text with PINE and I will send you a copy." An hour after that (still January 31), Gotterbarn notified the list: "The attachment seems to have some problems :-)"[13] While you are reading and commenting on the tome I sent, I will try to get this table straightened out (he said hopefully!)" Gotterbarn did not solve the problem until late Sunday (February 2). Early Monday morning he wrote PRFCMP-L:

> Sorry about the problems with the early version of the table. Only one person (with a MAC) was able to read it. To avoid all difficulties with versions or word processors etc. I spent a pleasant ground hog day (a special holiday in the colonies) rebuilding the table.

Tuesday morning (February 4), Mechler had some bad news for Gotterbarn (using PRFCMP-L): "In the second version of code relationships [comdes2.doc] the references do not line up with the correct code. It appears as spaces, tabs, etc. were not included. Nothing lines up

under IEEE or PMI. Norman quickly emailed Mechler directly: “Mine seems to be reasonably lined up; at least, readably so! Here it is.” What lined up for Norman did not line up for Mechler. After several more exchanges, Norman realized that the problem was not visible on his computer screen because of the font used. Everything lined up using the default font, Courier, but not using any font having variable spacing. He advised Mechler to print using Courier. “If that doesn’t work for you, let me know and I will get a printout with Courier and fax it to you.” But, Norman added, waiting for the fax might mean waiting till tomorrow because he would have to use his home PC:

> I do not have immediate [access] to the right combination of resources right where I am—Our VMS prints to a lineprinter, and the stuff in labs is occupied by hordes of students who always succeed in breaking things. We have the equipment, but to get a working combination of monitor, disk drive, and printer in the middle of the semester is quite an achievement here!

Norman had figured out what the problem was and had told Mechler—but not the listserv (or Gotterbarn). Norman and Mechler seem to have forgotten that they were working “off the list”—or, perhaps, it did not occur to either of them that Gotterbarn might also be working on the problem. But Gotterbarn was working on it and, on February 5 (Wednesday), he wrote Mechler triumphantly:

> The copy of the code you sent me lined up on my system, standard IBM clone. I checked with someone else here and they had no problem either, so I thought maybe something flipped going across the atlantic and I check[ed] with someone else in the states. They didn’t have a problem either. Let me know your fax number and I will send you a hard copy of the table.

Mechler sent the fax number, saying nothing about Norman’s discovery of the day before and, indeed, without any message at all. Did this unusual curtness mean that Mechler was too busy to chat (or that he had had enough chatting about the attachment)? While we cannot answer that question, we can be pretty sure that the reason Mechler sent the fax number was that Norman’s discovery had not solved his problem and Mechler wanted to see Gotterbarn’s table. Gotterbarn nonetheless had to put off sending the fax. The Went Building had (he soon learned) only one fax machine, requiring a code and authorization to use. Rogerson, who would have had to authorize Gotterbarn’s use, was out of town. So, instead of faxing, Gotterbarn kept working on the problem. By the next morning (February 6), he had remade Norman’s discovery (or, rather, confirmed a hypothesis Ben Fairweather, the philosopher, had posed).[14] Gotterbarn first wrote Mechler directly and then informed the listserv, adding new details:

> The table is pure ascii text table and must be printed using a mono-space font, something exciting like courier :-). Copy the table onto your word processor, select the table, select a mono-space font which does not cause the table to wrap. If there is nothing under PMI for 1.01 then the table is wrapping. This should work for most systems that I know of.

Norman confirmed a few minutes later (using the listserv): “Yes, I discovered that. Any of the courier fonts works fine with it in MS Word.” Early Friday (February 7), Mechler also confirmed (using the listserv): “Thanks everyone. I finally got a lined up copy of the matrix. Thanks Don.” It had taken exactly one week for three experienced software engineers (two of them still practitioners) to get comdes2.doc to print properly on Mechler’s equipment.

## 7.3 Comparing codes

The table was worth the effort. Below is the first part of it, the imperatives under Rule 1 and their sources (Rule 1 having the fewest sources). (For the entire table, with legend, see 7. Appendix.)

[Insert]

| | AAES | ABET | ABET-G | ACM | ACM-G | BCS-C | BCS-P | ICCP | ECPD | IEEE | NSPE | PMI |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Product | | | | | | | | | | | | |
| 1.0 1 | | | | 3.4 | | | | | | | | II.b |
| 1.02 | A | | | 3.4 | | | | | | | | II.b |
| 1.03 | C1 | B.2 2 | | | | 20 | 1.3 | 2.5 | | ‘6’ | ‘II.2.a' | I.b |
| 1.04 | | | 1.c.1 | 3.4 | | | | | | | | |
| 1.05 | | | | 1.2 | | | | | 2 | | | II.b |
| 1.06 | | | | | | | | | 2 | | | II.b |
| 1.07 | | | | | | | 2.4 | | ‘3' | | | II.b |
| 1.08 | | 1.c.1 | | 3.4 | | | | | | | 'II.2.b' | II.b |
| 1.09 | | 1.c.1 | 2.5 | | | | | | | | | II.b |
| 1.10 | | | | 1.8, 3.5 1.7 | 1.8 3.5 | 2 | 3.5 | | | | | I.b |
| 1.11 | | | | | 1.7 | 2 | 4.6 | | | | | |
| 1.12 | C2, C4 | A.1 | 1.a, 1.c.1 | | 2.3 3.1 | 1.1, 1.4 | 1.7, 8 | | | 8 | | |
| 1.13 | | | 4 .j | | | | | | 2 | | | |
| 1.14 | | | | 1.2 | | | | | | | | |

The rationale for choosing just these twelve codes is not obvious. All are American or British. Why none from the rest of Europe? Why none from Latin American, Australia, or Canada? All the *engineering* codes are American (unless, following the British, we count the BCS's as an engineering code). Why not even one (other) engineering code from Britain (say, the code of the Institute of Electrical Engineers)?

There is another oddity. There are two ACM codes where there should be one. The ACM-G (the commentary on the Code of Ethics and Professional Conduct) includes the entire ACM Code of Ethics (ACM). In this respect, the ABET Guidelines (ABET-G) differ from the ACM-G. Not only is it a document separate from the short ABET Code of Ethics for Engineers; it is also missing some parts of that code (specifically, the Fundamental Principles). Like the NSPE's code of ethics, the ACM code does not seem (properly) divisible into code and independent guidelines.

The codes divide neatly into six for engineering (AAES, ABET, ABET-G, ECPD, IEEE, and NSPE) and six for computing (assuming the Project Management Institute is primarily concerned with managing software projects). But two engineering codes seem odd choices: The AAES code (American Association of Engineering Societies, Model Guide for Professional Conduct), though adopted in 1984, was almost forgotten by the mid-1990s. It is one of those "failed codes" no one would want to copy. The ECPD "code" (Faith of the Engineer, Engineer's Council for Professional Development) is a one-page "creed" adopted in 1948 by what later became ABET. It was never intended as a code of ethics. The ECPD adopted its first (and very successful) code of ethics in 1947 ("successful" insofar as most major engineering societies adopted it within a few years of its publication).[15] ABET let The Faith of the Engineer (the "Faith") die quietly. What did Gotterbarn hope to achieve by citing it? He did not say. The obvious answer is political: give engineering the same number of codes computer science had. But perhaps there is another explanation. Both the "Faith" and AAES code appear (on facing pages) at the back of the Martin and Schinzinger text.[16] Their presence in that book may have misled Gotterbarn about their importance. He expressly cites that text as the source of "[m]ost of the [engineering] codes".[17] There is other evidence for this explanation: *all six* of the engineering codes Gotterbarn cites (and only those) appear in Martin and Schinzinger (1989). Gotterbarn's dependence on that text would also explain why no British code of engineering ethics is cited, though codes of the British Computing Society are.[18] The Martin and Schinzinger text includes no non-American codes.

If this explains Gotterbarn's choice of engineering codes, then we are entitled to draw two conclusions. The first is that neither Gotterbarn himself nor those with whom he was working at CCSR knew much about engineering codes. If engineers (and their philosopher friends) dominated "SEEPP\E", now computer scientists (and their philosopher friends) were dominating the work of the reorganized SEEPP. The second conclusion is that Gotterbarn probably lacked the time to find out what he did not know. He had a tight schedule. The only convenient library (DMU's) lacked a recent American text on engineering ethics. And there were then few, if any, websites displaying codes of ethics.

While I am also tempted to explain the absence of codes of ethics other than those from engineering and computing in the same way, I think there is a simpler alternative. Gotterbarn may have been interpreting the Joint Steering Committee's request (January 6, 1997) that he

"document the architecture of this code so that reviewers can understand the components, which parts are derived directly from other codes, and which parts are specific to software engineering" as *implicitly* asking only about computing and engineering codes. That is not unreasonable for two reasons. First, the Committee's immediately preceding request (its first) was even more limited ("complete a draft code of ethics and professional conduct that has its roots in codes currently in use *by other engineering professions as well as the codes of the IEEE and the ACM*").[19] Second, precedents closer to software engineering would naturally carry more weight than precedents more distant. Every clause had precedents close to software engineering. Gotterbarn had no reason to look further. He must not have known that until late in the process, though. Had he known that earlier, he would, presumably, have made clear that volunteers should only look at codes related to engineering and computing (and so saved Norman and Mechler the trouble of recovering notes on the AICPA code).

Whatever reason there is to criticize the choice of codes, the overall message of the table is plain (as even the part reprinted above suggests). Version 1 seems to correspond most closely to computing-related codes: ACM's and PMI's. That, no doubt, is—as Gotterbarn points out in his comments—because the imperatives under Rule 1 concern software (with which ACM and PMI are specifically concerned). Yet almost every imperative also has at least one "engineering hit". Contrary to what Frailey and Shaw argued, Version 1 was not, in substance, all that different from existing codes. It therefore is not likely to be less practical than these. The one way in which Version 1 seems to differ from the others should be an advantage. Version 1 covers more than any other single code does (though not more than all together). Gotterbarn could, it seems, have treated Version 1 as what Cabrera and Frailey had asked to have delivered by February 14, a "draft code of ethics and professional conduct that has its roots in the codes currently in use by other engineering professions as well as the codes of the IEEE and ACM." Nothing in the table, or in the comments so far received from the working group, required a total rewriting or restructuring (rather than a series of narrowly targeted amendments). What did seem to, if anything did, was the prejudice against Version 1 that the premature criticism of Frailey and Shaw may have generated within the Steering Committee (5.6).

Gotterbarn's table (the attachment) came with both a one-page single-spaced covering letter and (following it) a two-page single-spaced document bearing the very long title:

> Revision of
> DRAFT DOCUMENT RELATING
> COMPUTING, ENGINEERING, AND SOFTWARE ENGINEERING CODES.
> v2 30 Jan 1997
> The Architecture of the Draft Software Engineering Code of
> Ethics and Its relations to other Codes of Ethics

The authors of this document, "Gotterbarn & Rogerson", must have begun it at least a few days before January 30, but they must have revised it on January 30—after Mechler's annotations arrived. The third paragraph says that "the committee" (not "the working group" or "task force") "examined other codes of ethics including…"[20] The list of six items includes W.J. King's *The Unwritten Laws of Engineering*, the AICPA code, and the PA State Registration Board's code. The list is plainly Mechler's, but nonetheless differs from his list of eight in three ways. First, the

NSPE code is not there.[21] The reason for this difference seems to be that the preceding paragraph had already identified the NSPE code's structure as the model for Version 1. There was no need to mention it again so soon. Second, the Institute for Certification of Computer Professions (ICCP) Code of Ethics is not there (even though it is one of the twelve in the table). There seems to have been no reason to omit the ICCP code—except that it added little to the variety of sources already listed. Third, the list is not quite in Mechler's order. Mechler had the IEEE code first and the ACM code fourth. Gotterbarn & Rogerson moved the ACM code to first place. Alphabetical order does not seem to explain this change. If the order were alphabetical (as it is in the table of codes), the AICPAP code would come second, not fourth; the PMI code would not be third; and so on. The reason for putting the ACM code first seems to be political. So far, opposition on the Steering Committee had come from two ACM-appointed members (Frailey and Shaw). Putting the ACM code first here might "help the medicine go down".

Overall, the statement on "architecture" seems designed to introduce Version 2 (though the first use of "SECEv2" is not until the seventh paragraph). Thus, the first paragraph begins:

> This document was developed in response to a request for the IEEE-CS/ACM Joint Steering Committee for The Professionalization of Software Engineering. The architecture of the draft Software Engineering Code of Ethics (SECEv1) has been modified in the light of comments from the software engineering and computer ethics communities.

The fourth paragraph, after reporting that "the committee" had distributed SECEv1 to others for comment, says that "in light of those comments and further work by the committee, the structure of the code has been revised to reflect significant aspects of other engineering and computing codes." The remaining eight paragraphs describe SECEv2. So, for example, after noting that "[many] codes" such as the British Computer Society's and the ACM's include "a preamble describing the roles and functions of the code for the practicing professional", it indicates that "[the] structure of SECEv1 has been revised in accordance with this model".

Though most of the document on architecture follow closely Gotterbarn's early paper on functions (for example, using SECEv1 to illustrate his "three levels"), it does report two surprises. First, the original purpose of constructing the table ("documenting the architecture") had been to "show which parts [of SECEv1] derived from other [engineering and computing] codes and which parts are unique to software engineering." That purpose presumed that there are "clear lines of demarcation between engineering, computing, and software engineering." The table is "not consistent with that presumption." None of the imperatives is unique to SECEv1, and virtually every imperative has a counterpart both in some engineering code and in some computing code. Mechler had been right. (5.6) Second, there is nonetheless an interesting contrast between engineering codes and computing codes: "At the moment some imperatives are contained mostly in engineering codes (the outer columns of the chart) and [some] imperatives are contained mostly in computer codes (the inner columns of the chart)." Those imperatives sparsely represented in both engineering codes and computer codes concern "management issues" or "very specific…technical issues in software engineering."

The covering letter (January 31, 1997) preceding the document on code architecture began with thanks to "those who commented on my draft on Functions of Codes of Ethics and

who contributed to annotating the code…especially Ed." This explicit thanking of Mechler seems right, but it does suggest two questions. First, who else contributed to the comments or annotations?[22] Second, what in fact was Mechler's contribution? What did Gotterbarn mean when (in paragraph 2), he said he had "integrated Ed's results into the baseline document [the table]"? We can be confident that someone other than Mechler did contribute. Mechler had provided annotations for five codes; the table covered twelve. Norman's annotations did not arrive in time (and would, in any case, not have fit the engineering-computing plan). Who did the work for the other seven codes? The answer may be: Gotterbarn. His general practice seems to have been to thank explicitly anyone who did any significant work. He had special reason to thank some others if he could, for example, his host, Rogerson. We must therefore suppose Gotterbarn to have analyzed the seven codes more or less by himself.[23] What else must he have done?

We know that Mechler found analyzing five codes to be taxing, even when the work was spread over several weeks. We must then assume that Gotterbarn spread his work over several weeks as well—or, at least, did not do everything in the twenty-four hours between the time he received Mechler's annotations and the time he sent the table out. But, if Gotterbarn worked on the table for several days or weeks before Mechler's annotations arrived, he was unlikely to have annotated just the codes Mechler omitted. Gotterbarn did not know what Mechler was doing, or even that Mechler was doing anything, since Mechler did not tell him. Gotterbarn, in turn, did not tell Mechler there was a deadline for the work if Mechler (or other contributors) wanted their work to be of use; and, indeed, there may have been none (apart from February 14). Gotterbarn seems to have worked as fast as he could; the table was just one of several substantial projects soon to be revealed. So, by January 30, he must have had an almost complete table for all twelve codes. He would have included the codes Mechler did (where there was overlap) for the same reason Mechler included them: they were too important to omit (and, except for the ACM, PMI, and ICCP codes) were included in the Martin and Schinzinger text on which Gotterbarn relied). So, the arrival of Mechler's annotations must have been a complete (and unique) surprise. Their late arrival meant that they could provide no more than a check against Gotterbarn's own work.[24] Had Mechler been a little slower, his work would have been entirely wasted (as Norman's was). Gotterbarn was no longer moving slowly, waiting for others to do something. Those used to his patient style did not, and probably could not have, anticipated what Gotterbarn did in January—or what he would do over the first two weeks of February.

The rest of Gotterbarn's January 31 letter was a status report. Paragraph 3 promised to "[revise] the Functions paper in light of your comments". Paragraph 4 reported that Gotterbarn "[is] fortunate to be at the Centre for Computing and Social Responsibility this semester, which is supplying both intellectual and clerical support of this effort. Paragraph 5 lays out a "tentative [six-step] plan" (without dates): "1. [Submit] a revised code to the steering committee for review and comment, 2. Revise the code in light of the comments, 3. Print the Code in the ACM and IEEE publications with a survey asking for comments and voting on each item", and so on. The paragraph ends with the promise that a "more detailed plan is forthcoming".

### 7.4 The plan to revise

On February 4, 1997, Gotterbarn addressed a sixteen-page (single-spaced) memo, "Dear EC (executive committee)".[25] The "executive committee" had just three members: Gotterbarn, Miller, and Rogerson. The executive committee formalized what had become Gotterbarn's practice over the two weeks preceding—working closely with Miller and Rogerson, both of whom responded quickly to calls for help and seemed to Gotterbarn to know what they were doing. Why Gotterbarn thought it necessary to formalize his "kitchen cabinet" is a mystery. As a formal body, the executive committee looks unbalanced: three relatively senior academics, all computer scientists without any background in engineering.[26] Though all had considerable experience in industry, that experience would not have been obvious to anyone looking at affiliations (and, in any case, was experience as "software engineers", not engineers strictly so called). Since Rogerson was not an American, he gave the EC an international flavor; and since he was in information management, he might also seem to offer a management perspective (as indeed he did). But Miller was almost Gotterbarn's twin: an American professor of computer science.

The first paragraph of the February 4 memo explains that what follows is "a copy of the code" (Version 1) with comments integrated. The source of each comment is indicated by initials: DF (Frailey), MS (Shaw), ED (Mechler), and so on. Most of the commentators are familiar from Chapter 5, but two are new: sr (Rogerson) and DS ("second pass comments" from Gotterbarn and Rogerson—"D" for "Don" and "S" for "Simon"). Some of the comments "raise difficulty" because "they are based on a miss-reading of the code" or are "just plain wrong" (for example, Shaw's view of "bribery as good international business practice"). The task of the EC is "relatively quickly" to restructure the code "a. to improve it, b. to address objections raised, and c. [to] put together changes suggested with reasons for each and circulate to the task force." Gotterbarn then sketched his own ideas for improving the code:

A.preamble introduction.

i. revision of the introduction to include what we consider main functions of code
ii. indication that code is not a list, black letter law, as in BCS code—put in preamble
iii. make clear structure based on relationships[27]
iv make clear that client & employer may not be synonymous, may even mean professor and student.
v. include all people related to SE, like academics
vi. YOUR ADDITIONS OR DELETIONS FROM THIS LIST

B. bigger issues

i. need to clearly bring QUALITY into the code—maybe in places that mention efficiency, for example 1.13 replace "productivity" with "quality"
ii need to bring in teamwork and management issues
iii. a professional owes higher order level of care
iv. relate all stakeholders not merely "end users"
v. address "intensity issue" by replacing "assure" with different terms
vi. YOUR ADDITIONS DELETIONS FROM THIS LIST

C. minutia

i. change "RULES" to "PRINCIPLES"

ii. will we interleave examples 1.1,1.2 into list of principles, NSPE model or make the examples a separate list of guidelines ACM model.
iii. YOUR additions or deletions from this list

Clearly, Gotterbarn has gone beyond resolving the issues that Frailey, Shaw, and members of the task force had raised the preceding autumn. Who, for example, had asked whether "client" might mean "professor" or "student"?[28]

Even the two (substantive) "minutia" suggest that Gotterbarn was planning to rewrite the code in ways that went well beyond the criticism received in October and November. Both the "minutia" are important changes in architecture. The change from "rules" to "principles" is not merely the substitution of a three-syllable word for a one-syllable one. It also substitutes a more ambiguous term for a less ambiguous one. I had called the general standards "rules" to stress that they were as binding as the particular standards, merely more general forms of the same sort of standard; I had used "shall" in them rather than "should" for the same reason. For many people, "principle" suggests a standard of a different kind from "rule", for example, a standard that should have weight in deliberation, not—like "rule"— one that, while open to interpretation, is binding, something one either obeys or violates.[29]

While Gotterbarn seems unaware of this possible understanding of "principle", he would soon (February 10) offer his own suggestion for distinguishing between "rule" and "principle": "these [the Rules] are not really single rules but statements that embody a set of activities. This change [to Principles] lessens the possibility of misreading of the code as a checklist." Since all rules embody a set of activities, Gotterbarn has not actually explained the distinction. He has simply offered the name "principle" as a better way than "rule" to lessen the likelihood that the standards in question will be used as a checklist. What reason he had to believe that the change of name would actually accomplish that end he did not say.[30]

Similarly, segregating the particular standards ("examples") in a separate section, as the ACM code does, might (as Gotterbarn also claims) lessen the likelihood that the Principles will be used as a check list. But it might not. Other outcomes seem equally probable. For example, a separate section for the principles might invite hasty users to forget about the particular standards altogether or suggest that the particular standards be treated as mere commentary rather than as binding rules, something to look at only if one has doubts about what a principle on the "check list" means.

That Gotterbarn would describe such architectural changes as "minutia" is surprising. The "minutia" in question might well make the difference between a code widely used and one soon forgotten. Gotterbarn's later extensive reworking of the code, whatever its merits, seems to show that he did not regard these issues as "minutia". Perhaps "minutia" is no more than an unfortunate "slip of the pen" (or whatever the word-processing equivalent is). That Gotterbarn would make empirical claims about how the code is likely to be used without evidence is not surprising (even if interesting). Though we know little about how codes of ethics are used, claims about the heuristic advantages of writing a code one way or another are common (and, in my experience, often wrong—or, at least, implausible upon examination).[31]

7.5 The great revision

The next few days would have been hard for Gotterbarn even without trying to straighten out Mechler's table. Gotterbarn, apparently with much help from Rogerson, rewrote the entire code (drawing on responses received in October and November), annotated the changes, sent the annotated revision to Miller and Rogerson for comment, and revised accordingly. Since Miller was in Illinois, not working down the hall (as Rogerson was), we have some idea when Gotterbarn completed Version 2. On the morning of February 10, Miller sent in his last comments (responses to Gotterbarn's replies to his earlier comments). Some reveal how important Rogerson was. For example, Miller had found several provisions with "outwith" in them and confessed to not knowing what the word meant. Gotterbarn had responded, "Simon's contribution—a scot[t]ish idiom, meaning 'standing outside of'. He thought it would add an international flavour to the code and determine if anyone read this far." Miller's February 10 reply is: "HA, HA… I passed the test! International flavor or international obscurity?"[32] Writing a code of ethics, though serious work, need not be solemn.

Some of Miller's comments reveal the process of choosing words by which a code comes into being. For example, Miller had commented (on the always troublesome) 1.14 ("Avoid fads…"): "Are you going to include a citation to a handbook of 'standard' software engineering practices. I want that reference." Miller's request for a reference was, of course, ironic. The Software Engineering Institute was, in effect, working toward such a handbook piecemeal. Any handbook was (as Miller knew) decades away (and, like engineering handbooks, would never be complete or entirely up to date). The Body of Knowledge Task Force would, at best, offer only relatively uncontroversial standards as part of the "body of knowledge" all software engineers share (a basis for identifying those who know the "basics" rather than for guiding the experienced). Some practices might, though popular, be novel enough not to belong in the body of knowledge or even in SEI standards and yet not be what anyone would understand as a fad. Gotterbarn therefore answered Miller:

> We have struggled with that one…We received major attacks on "fads".
>
> Perhaps answer your "standards" question in terms of "what is publicly defensible" or in terms of results from the skills task force. I know neither of these is a good answer, but they are answers other technologies would use in defending that a practice was 'standard'.
>
> Any ideas how to reframe this point?

Miller's February 10 suggestion is to rewrite 1.14 to read (something like): "software engineer should follow industry practice that, in his/her judgment, are most appropriate for the task at hand. Novel and experimental techniques may be appropriate, but not for the sake of novelty."

That evening, barely a month after he arrived in Leicester, Gotterbarn sent the task force (an annotated) Version 2 of the "Code of Ethics for Software Engineers". Having had enough trouble with attachments when he sent out the table of code references, Gotterbarn did without attachments, dividing his message into three emails (none large enough to clog a standard mailbox). The first was a page-long covering letter with "Principles 1-2"; the second, "Principles 3-4; and the third, "Principles 5-7".

The division into parts did not work quite as planned. Gotterbarn sent off his emails just before 7:00 PM Leicester time. By 6:00 PM (EST), that is, several hours later, Norman used the

listserv to tell Gotterbarn he had all three (February 10, 1997).[33] Next morning, however, Milton Fulghum (Missouri) emailed (through the listserv): "I have received parts 1 and 2 but not three." About noon (February 11, 1997), Langford (Kent, two hours by train from Gotterbarn) sent a similar response to Norman (through the listserv): "O fortunate one—I managed to get (2) and (3), but am still anxiously waiting for (1)…" Late that afternoon, Gotterbarn advised "patience…[the] net being what it is, the messages are taking interesting routes to get to you all." He gave one example to make the point:

> I mailed them [the three emails] from the Centre for Computing and Social Responsibility here in England. The director [Rogerson]—office across the hall from me—did not get part one until 2:00 pm this afternoon….[while] a research associate [Fairweather] upstairs had all three this morning (GMT).

Gotterbarn urged everyone to "tackle, grapple with, the parts you do have :-). I appreciate your enthusiasm, but we are actually slowing down their arrival filling the net with these messages (me too, I guess with this message)."

By the next morning, everyone who had any part of the tri-partite message had all three. But there was at least one problem of another kind, the mailing list itself. Weil emailed Gotterbarn (February 12, 1997): "I don't understand why my colleague Michael Davis has not received the recent flood of e-mail. We thought csep@charlie.cns.iit.edu was on the [list]." Gotterbarn responded that "[Davis] has always been on the list as weil@charlie.acc.itt.edu [sic] and is still on that way."[34] He recalled "we were having some trouble with his address last year and…per instructions we changed his address to yours, only at .acc." Gotterbarn then promised to change the address if it was now not working. He went on to say that he "would appreciate his comments and yours on the current version of the code and our plans"[35] and concluded by asking "Are you still tracking us for NSF?"[36]

Once complete, the tri-partite message was an impressive, even if incomplete, document. It was seven single-spaced pages long. Many of the code's seven "Principles" or specific clauses had bracketed numbers inserted (1-19). Each number corresponded to a comment at the back of the email (with numbering beginning at 1 in each email). Each number either marked a change already made or identified a place where a change was "under construction". The comments explained a change and sometimes provided a justification (though some of the Principles also had explanations preceding them). There was no "Introduction" because "changes to the Preamble are being worked on now."

The covering letter (February 10, 1997) not only explained these matters and urged everyone to read through the draft "in the next few days" but to "give us your comments" and "your suggestions for improvements". The letter also included two substantive paragraphs explaining Version 2's architecture. (These paragraphs used the future tense, suggesting that they had been pulled from an earlier planning document without editing.) The first described "v2.0" as having an architecture "similar to version 1.0." It "will start out" with a "preamble or introduction explaining the code and [providing] guidelines on its use."[37] Following the preamble will be "a series of keyword principles describing a software engineer's obligations in a variety of relationships." (Apparently, what Gotterbarn meant by "keyword principle" is what most people would call a "title" or "keyword".) Associated with each keyword is a "high level

description of the software engineer's obligations [what Gotterbarn elsewhere calls a 'principle'].” Following each of these is “a series of clauses detailing some of the obligations under this Principle.” These obligations “are taken from all three levels of obligations in a professional code of ethics” (human, profession as such, and software specific). Gotterbarn had *not* reworked the Code to mirror the three levels.

The second paragraph calls this architecture of “major principles followed by explanations or examples…fairly common.” It then gives two carefully chosen examples to prove the point: “it is used by the NSPE and the British Computer Society”. The mention of the British Computer Society protects Version 2 from being treated as an engineering-style code rather than a computing-style code (or as an American-style code as opposed to an international code). While politic, the claim that Version 2.0 resembles other codes in this way is only very roughly true. Version 2.0 (and its predecessor) are alone (at least among the codes Gotterbarn discusses) in *explicitly* treating the particular rules as examples (special cases) of the general rule under which they are listed. In most codes having particular rules listed under a general one, the relationship is left to interpretation. Also “quite common”, according to Gotterbarn, is the use of the preamble to inform users that “the code should not be considered as a complete list of professional ethical obligations”, for example, “all codes in the architecture table follow this format”. In fact, that is not true of any of the engineering codes cited—though it is certainly reasonable to interpret them as having such a provision “implicit”.[38] Version 2.0 (and its predecessor, 1.0) had, in this respect, joined a relatively small number of recent codes (of which the ACM and BCS codes are two important examples).

Code 2.0 (and, presumably 1.0 as well) “differs from most codes” in “its use of keyword principles”. Most codes do title their major sections (though some, like the 1979 IEEE code) merely give “reference titles” such as “Article I”, “Article II”, and so on. Some, such as the Australian Computer Society Code of Ethics, also have “keyword” titles (such as “Priorities”, “Competence”, and so on) and general standards (such as “I will serve the interests of my clients and employers, my employees and students, and the community generally, as matters of no less priority than the interests of myself and my colleagues”) as well as particular rules under them (for example, “I will endeavour to preserve the continuity of computing services and information flow in my care”).[39] The “SE code differs [from other codes, according to Gotterbarn,] in that its keywords are tied to professional relationships”. This is important because “[works] on computer ethics have found this organization [by professional relations] useful in covering the topics of professionalism and useful as a nmonic [mnemonic] device.”[40]

The comments suggest that Version 2.0 is, overall, a less demanding (“more aspirational”) document than Version 1. In many respects, that is true.[41] So, for example, comment 6 notes that “Strive to” has replaced “Assure that they” in 1.02 (“Strive to understand fully the specifications for software on which they work”) to “placate those who appeal to impossibility of precise specs”; and comment 10 notes that “Aspire to” has replaced “Assure that they” in 1.12 (“Aspire to identify, define and address ethical, economic, cultural, legal, and environmental issues”) as part of a general program to use a “more aspirational and intentional phrase in these level 1 type obligations in 1.10-1.14.”[42] Again, 2.02 (“Approve software only if they have a well-founded belief that it is safe, meets specifications, and has passed appropriate tests….”) is the result of three amendments: One substituted “well-founded” for “well-documented”. The second deleted the word “all” before “appropriate tests”. The third substituted

"appropriate" for "proper". Note 15 points out the first change but does not explain it. For software engineers bedeviled by lack of documentation, the substitution of "well-founded" reduces Version 1's overall emphasis on documentation. A programmer need only think she has good reason to approve; she need not document her reasons. When a question arises a year later and she is gone to another job (or forgotten what she thought at the time), how will anyone know what her belief was—what she considered "safe", what she thought the specifications were, or what tests (if any) she thought "appropriate"? Similarly, note 15a explains that "all" was deleted "to avoid the suggestion that exhaustive testing is required" (just what Mechler thought had been avoided by having "as appropriate" precede all clauses under Rule 1). But, in avoiding the "suggestion" of exhaustive testing (a standard generally too high to be practical), the deletion of 'all' had the effect of permitting a software engineer to do no more than *some* appropriate testing, something short of all the tests that should (together) be done.[43]

Though the overall tendency of the proposed revisions was to make the code less demanding (more "aspirational"), some changes seem to add to its demands. For example, Rule 2.05 ("Be fair and truthful in all public statements") had become "Be fair and truthful in all statements, particularly public ones, concerning software or related documents." Comment 18 explains that, as originally written, 2.05 "indicated that it was ok to lie in non-public statements" (on the questionable but not unknown interpretive principle that whatever is not expressly forbidden is allowed).[44] Comment 18 apparently overlooks what the code's introduction says about the status of particular rules. They are instances of the general rule. So, for example, 2.05 is an instance of "2. PUBLIC Software engineers shall, in their professional role, act only in ways consistent with the public safety, health and welfare." Any non-public statement inconsistent with the public safety, health, or welfare is also disallowed. What the revised 2.05 seems to disallow (that the original did not) are non-public lies concerning software or related documents that are consistent with the public safety, health, and welfare. For example, 2.05 now seems to disallow lying in non-public statements to tyrants or unscrupulous employers even when such lies serve the public safety, health, and welfare. A software engineer must be "truthful" even with them. Since Principle 2's "In particular" has no "as appropriate", Version 2.0 therefore sets a higher standard than Version 1.

Some changes that seem intended to raise standards may in fact lower them. Consider, for example, comment 8 under Principle 3 (Judgment). The comment notes the need for some "social clause" in 3.06 ("Participate in no decision of a governmental or professional body, as a member or advisor, concerned with software, or related documents, in which they, their employer, or their client has a financial interest"). "[As] this clause stands judgment is merely a technical thing and does not involve social issues." The comment then offers the following as possible substitutes for 3.06 (apparently intending to raise the standard):

> Only make judgments which can be publicly justified as being in the best interest of quality of life, and the environment.
>
> All technical judgments should be tempered and guided by the need to support and maintain human values.

> Be aware that all technical judgments impact other human beings. They have stakeholders other than employers, clients, and users.

Gotterbarn seems to have understood that 3.06 (and, indeed, all of Principle 3) is concerned with maintaining independent professional judgment (that is, avoiding conflict of interest), not with substantive issues discussed elsewhere in the code.[45] What he does not seem to have understood is why it is important to have a separate section on technical judgment after two sections ("Product" and "Public") concerned with the very substantive issues he proposes to incorporate into 3.06. The avoiding of adverse financial interests is a *means* of helping to ensure that the judgments in question "can be publicly justified as in the best interests of quality of life" and are "guided by the need to support human values". Substituting the general imperative for a particular means, allows the end to be pursued by other means instead (for example, by giving to the poor some of the profits derived from keeping the adverse financial interests). The point of 3.06 is that there is no substitute for avoiding this sort of conflict of interest. Good intentions (and even compensatory good deeds) are not good enough. Gotterbarn's recourse to good intentions as a substitute for the flat prohibition of certain conflicts of interest is thus a weakening of the code even though intended to strengthen it.

Some changes are hard to categorize on the dimension of more-or-less demanding because their overall effect is hard to gauge. For example, Version 2.0's Principle 1 has "free of error" where Version 1 had "free of significant error." The comment on this change explains that "[we] felt uncomfortable with the phrase…It might be used as an escape clause—'I had lots of errors but didn't think any of them significant'. "We" tried to compensate for this strengthening of the code by specific changes in Principle 2 ("PUBLIC Software engineers shall, in their professional role, act only in ways consistent with the public safety, health and welfare.") But there "is still some concern with this change [deleting "significant"] because it is not possible to write error free code. Any suggestions for balancing these concerns?" The overall effect of this combination of changes may or may not be a change in what is actually demanded. What seems clear is that Gotterbarn, Rogerson, and Miller ("we") are aiming at something very close to the balance that "free of *significant* error" was intended to achieve. (Eventually, they would seek to achieve that effect by radically different language.)

Other changes are hard to assess on the more-or-less demanding dimension for other reasons. For example, the third change in 2.02 substitutes "appropriate" for "proper". The explanation for this change (here and elsewhere in the code) is that "appropriate" is "less vague" than "proper". There are "technical answers to the question 'which methods are appropriate?'"—but, apparently, not to the question, "Which methods are proper?" Many readers may find the claim that "appropriate" is less vague than "proper" novel and suppose that all that has happened is that a "$3 word" has replaced a shorter one. Others may suppose that technical standards set what is "proper" but interpretation (professional judgment) must decide what is appropriate. Given that supposition, the change from "proper" to "appropriate" would be a weakening of the code (a substitution of individual judgment for common standard) rather than (as the comments claim) a strengthening. Some word must be chosen, and all will be open to misinterpretation. The most precise word is, all else equal, the best. But what makes a word the most precise? And how are the drafters of any code to know?[46]

Another example of hard-to-assess changes is the general replacement (in Principle 1) of "assure" with "ensure" (where "assure" is not replaced by "strive" or some other less demanding term). But here the claim to know which word is better relies on a linguistic authority (the *Oxford English Dictionary*). Comment 3 (under Principle 1) defends the replacement as a weakening of the obligation: "'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."[47] Appeal to the OED may seem parochial. An American who looks up "ensure" in Webster's (or one of its American competitors) will find the entry under "ensure" to be: "same as *insure*" (a formal legal guarantee); or "make sure or certain" (something distinctly stronger than "strive to make happen").[48] So, in the US at least, replacing "assure" with "ensure" may not weaken the obligation (or even avoid the hint of formal legal guarantee). If anything, the replacement strengthens the language, substituting the robustly objective "ensure" for the faintly psychological "assure". And, in fact, I had chosen "assure" over "ensure" because I did not want to suggest "insure". I too was concerned to avoid the suggestion of "formal legal guarantee". What I had overlooked is that code-writers almost universally prefer "ensure" (even if they do not know why). Here, as in many other places, there is agreement about the end in view in the choice of a certain word but disagreement about the meaning (or likely impact) of the word in question. More, interesting, though, is that OED entries for "ensure" and "assure" are similar to Webster's.[49] The OED does not in fact appear to explain the preference for "ensure". Did anyone on the EC actually check the OED?

Amid all these changes in wording was one big structural change. The third part of the three-part email proposed to add a whole new section, "Principle N Management" ("N" being a placeholder until the new section received a number determining its place in the code). The principle itself was only partly stated: "A software engineer in a management or leadership capacity shall…." But under this half-stated principle are nine clauses (N6.06-N6.15). Most, such as N6.06 ("Assure that employees are informed of standards before being held to them"), are simply pulled from the clauses under Version 1's Principle 6 ("Colleagues"). Some are slightly revised. For example, N6.13 differs from Version 1's 6.13 only in having "remuneration" in place of "compensation" ("Offer only fair and just remuneration"). (The explanation for this change is that "in Europe ['compensation'] means 'an item given to redress a wrong done you'".)[50] One clause is entirely new ("N6.15 Not ask an employee to do anything inconsistent with this code"). Preceding the particular clauses is a warning that they are "under construction". Clauses yet to be formulated should include "enable, encourage, follow good practice", "provide direction to good software engineering practice", and something about "Team building, Communications, and Good practice". These clauses never entered any version of the Code..

The idea for a distinct section on supervisors, managers, or leaders may have come to Gotterbarn from one (or more) of five sources. First, Gotterbarn understood the code to be organized according to "relationships". Management is an important relationship, one software engineers regularly find problematic. Insofar as Gotterbarn's understanding of the code's architecture is the source of Principle N, Principle N is evidence that theory can make an important contribution to writing a code (by calling attention to an option that we might otherwise overlook). Second, Gotterbarn may have gotten the idea for Principle N from the ACM Code. Unlike most other professional codes, it has a section on "Organizational Leadership

Imperatives". While Gotterbarn may have gotten the general idea from the ACM Code, he does not seem to have drawn anythin more specific idea from it. None of Principle N's clauses seem to share language with any of the imperatives in the ACM code's (large) leadership section. Third, another possible source of Principle N is Rogerson. He had a special interest in the ethics of technical management. Fourth, the list of rules under Principle 6 had become significantly longer than any other. It therefore invited the question: can we divide this in some way? Fifth, and most probable, Gotterbarn may have gotten the idea from Milton Fulghum, a SEEPP volunteer. Fulghum suggested dividing Rule 6 in just this way in his comments of October 22, 1996. Among the computer files Gotterbarn retrieved from George Washington University after his arrival at DMU was Fulghum's email.[51]

7.6 Why the new Gotterbarn?

Just two days before the deadline Cabrera and Frailey had set Gotterbarn to deliver "a *complete* draft of the code of ethics and professional conduct", Gotterbarn was sending off for comment a draft much rougher than the one he had started with—a code lacking a preamble, differing in many important ways from its predecessor, with some provisions still "under construction, and the whole untested by outside comment"—, a code no more "[rooted] in the codes currently in use" than its predecessor but now salted with question marks.[52] Once a distant administrator, punctuating long silences with a new plan others were to follow and then waiting quietly for them to follow it, Gotterbarn had suddenly become active, inventive, and daring. He was doing much of the work himself. He was not simply revising the code to meet objections but rewriting the code as he thought it should be written. He was betting that he understood the politics of the Steering Committee, and of the two organizations that had joined to create it, well enough to undertake fundamental revisions at what must have seemed rather late in the game.[53] What explains this change in the way Gotterbarn worked? We may identify at least three (more or less) related possible explanations (in addition to having the more flexible schedule of a sabbatical leave).

First, Gotterbarn seems to have begun his co-chairing of SEEPP somewhat in awe of the IEEE-CS. At his first meeting with Barbacci and Melford, he had accepted the IEEE's standard-setting procedure, the organization by task force and working group, and even a list of working groups having little relationship to his previous experience helping the ACM write its code of ethics. That, in the mid-1990s, a computer scientist would, all else equal, defer to engineers on how to organize a large-scale project is understandable. The reason for coining the term "software *engineer*" almost three decades earlier was precisely to point up the superior organizing ability of engineers, to invite programmers, computer scientists, and the like to learn from engineers how to organize a large project, how to complete it on time, within budget, and as specified. In addition, the joint project had begun as an exclusively IEEE-CS undertaking. ACM had joined after the Steering Committee had begun work. Less clear about what it wanted the project to accomplish, the ACM had not demanded that its procedures be followed. Gotterbarn had begun work in what must have seemed an engineering environment. He needed time to learn his way around, to see the limits of the IEEE method, and to gain the confidence of the IEEE-CS appointees on the Steering Committee.[54]

Second, once organized, SEEPP could only work (Gotterbarn supposed) if he and Melford agreed. Gotterbarn could not act on his own (except as leader of his working group). Melford could prevent Gotterbarn from acting simply by falling behind in answering his email. He had a (unintended) "silent veto". While Gotterbarn accepted the IEEE's standard-setting procedures, including the division of labor among working groups, and Melford had time to meet, the requirement that the co-chairs agree did not matter much. SEEPP was able to produce the documents that the IEEE procedures called for—at something like the rate typical of the IEEE standard-setting process. However, once Melford's business began to overwhelm him, leaving him less and less time for outside activities, his "silent veto" became important. However much Gotterbarn doubted the value of the complex structure to which he had initially consented, it was now something he could not change unilaterally. He could not even (he supposed) properly email SEEPP, much less call a meeting, without Melford's express consent. He was like a fish in a frozen lake. When Melford resigned as co-chair, the lake thawed.

Third, Gotterbarn could work on his own through his own group, Professional Competence, but he could work there only within the group's limited mandate. His working group could not write a code of ethics (or a code of professional practice), only a part of it ("avoid generalization of expertise, keep current, keep staff current, truth about skill, what should be produced, appropriate knowledge base").[55] What Gotterbarn did ask, which was all that he could ask, did not much interest many of the volunteers. They slowly drifted away. Though he could see what was happening, he could (it seemed) do nothing about it. As SEEPP's co-chair, he had to be careful about organizational etiquette. If he ignored Melford, or just offended him, he might alienate IEEE-CS, risking the entire undertaking. Mechler succeeded where Gotterbarn failed precisely because he ignored the organization in a way Gotterbarn could not. Or, rather, Gotterbarn could succeed within the structure he had accepted only by using a device familiar to software developers, the "skunkworks", that is, "a group of people who, in order to achieve unusual results, work on a project in a way that is outside the usual rules."[56] Had Mechler not rushed in on his own, Gotterbarn would have had to invent him. Once Melford resigned, there was no need for a skunkworks. The Steering Committee, having learned to trust Gotterbarn, gave him the power to create a structure within which he could work. He then worked more or less as Mechler had, though within the rules and with a speed even Mechler never exhibited.

## 7. **APPENDIX: Comparison of Version 1.0 with Other Codes**

From: IN%o"PRFCMP-L@UTKVM1.UTK.EDU" "Professional Competence Standards Task Fo rce" 3-FEB-1997 07:10:33.22 To: IN%"PRFCMP-L@UTKVMI.UTK.EDU" "Multiple recipients of list PRFCMP-L" CC: Subj: Text version of table

Return-path: <@uga.cc.uga.edu:owner-prfcmp-1@UTKVM1.UTK.EDU> Received: from uga.cc.uga.edu by minna.cns.iit.edu (PMDF V5.1-4 #19448) with SMTP id <O11EYYFUVB5C8ZK3P4@minna.cns.iit.edu> for CSBURNSTEIN; Mon,

Sorry about the problems with the early version of the table. Only one person (with a MAC) was able to read it. To avoid all difficulties with versions of word processors etc. I spent a pleasant ground hog day ( a special holiday in the colonies) rebuilding the table.

Explanation of the table: The first column refers to sections of the SECvl. Only those items which closely matched the statements in SECvl are referenced. If the referenced item in the other code is almost a direct quotation, then the reference appears in single quotes.

The codes used for comparison, listed in alphabetical order, are: The American association of Engineering Societies, Model Guide for Professional Conduct(AAES); Accreditation Board for Engineering Technology's, Code of Ethics for Engineers (ABET C of E) and Guidelines for The Fundamental Cannon of Ethics (ABET G); The Association of Computing Machinery's Code of Ethics(ACM), and Guidelines for Professional Conduct(ACM G), The British Computer Society Code of Conduct (BCS C of C); The British Computer Society , Code of Practice(BCS C of P): The Institute for the Certification of Computing Professionals (ICCP); The Engineer's Council for Professional Development, Faith of the Engineer(ECPD Faith); The Institute of Electrical and Electronics Engineers, Code of Ethics(IEEE C of E); The National Society of Professional Engineers, Code of Ethics for Engineers (NSPE C of E), and the Project Management Institute "Code of Ethics for the Project Management Profession" (PMI). Several of these codes did not contain section and paragraph numbers, so the following reference procedure was followed. If the document was not divided into sections, its paragraphs were simply numbered sequentially starting with 1. If the code was divided into sections and paragraphs. The paragraphs were given an alphabetical designation and the paragraphs within each section were numbered sequentially starting with 1. Most of the codes are in Ethics in Engineering, Martin and Schinzinger,
and the other codes are on the NET Reference Table Comparing Software Engineering Code v 1.0 to Other Codes of Ethics. version 1.0 January 1997 Software Eng. C of Ev1:

| | AAES | ABET | ABET-G | ACM | ACM-G | BCS-C | BCS-P | ICCP | ECPD | IEEE | NSPE | PMI |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Product** | | | | | | | | | | | | |
| 1.0 1 | | | | 3.4 | | | | | | | | II.b |
| 1.02 | A | | | 3.4 | | | | | | | | II.b |
| 1.03 | C1 | B.2 2 | | | | 20 | 1.3 | 2.5 | | ‘6’ | ‘II.2.a' | I.b |
| 1.04 | | | 1.c.1 | 3.4 | | | | | | | | |
| 1.05 | | | | 1.2 | | | | | 2 | | | II.b |
| 1.06 | | | | | | | | | 2 | | | II.b |

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1.07 | | | | | | | 2.4 | | ‘3' | | | II.b |
| 1.08 | | 1.c.1 | | 3.4 | | | | | | | 'II.2.b' | II.b |
| 1.09 | | 1.c.1 | 2.5 | | | | | | | | | II.b |
| 1.10 | | | | 1.8, 3.5 1.7 | 1.8 3.5 | 2 | 3.5 | | | | | I.b |
| 1.11 | | | | | 1.7 | 2 | 4.6 | | | | | |
| 1.12 | C2, C4 | A.1 | 1.a, 1.c.1 | | 2.3 3.1 | 1.1, 1.4 | 1.7, 8 | | | 8 | | |
| 1.13 | | | 4 .j | | | | | | 2 | | | |
| 1.14 | | | | 1.2 | | | | | | | | |

| | AAES | ABET | ABET | ACM | ACM | BCS | BCS | ICCP | ECPD | IEEE | NSPE | PMI |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Public** | | | | | | | | | | | | |
| | C.3 | | | | 1.1 | | | | 3 | 1 | | |
| 2.01 | C.9 | Al, A2 | 1.b, 1.c.3 3.b | 1.3 | 1.2 2.5 | 7 | | 2.7 3.7 | 3 | ' 1' | II.1.a | IV.a |
| 2.02 | | | 1.b | 1.2 | | 1 | | | | 1 | II.1a | IV.a |
| 2.03 | | | 1b, 2.c | | 3.2 | 21 | 1.3 | 2.7 | | | 'II. 1.b' | I.b |
| 2.04 | | | | | 1.2 | | | | | 7 | | |
| 2.05 | C.3, | B.3 | 1.b, 2.c, 3.b | | 2.4 3 | 17 | | 3.4 | | 3 | I.3, 'II.3.a' | |
| 2.06 | A,C9 | B1, B3 | 1, 1.b | | | | | | 3 | | II.1.a | |
| 2.07 | C.8 | | | | 1.2, 2.4, 2.6 | 21 | | | 3 | 1 | I.1 | I.a |

| | AAES | ABET | ABET | ACM | ACM | BCS | BCS | ICCP | ECPD | IEEE | NSPE | PMI |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Judgement** | | | | | | | | | 2.1 | | | |
| 3.01 | | A.2, | 2.b | 2.5 | 2.5 | | | | | | II.3.c | |
| 3.02 | | A2, B4 | | 2.6 | | 7 | | | | '4' | II.5.b | |
| 3.03 | C.5 | A.2, | 4.c, 4.e | 2.6 | | 7 | | | | | 4 II.4.c, II.5.b | III.d |
| 3.04 | C.5 | A2, B4 | 4.c | 2.6 | | 7 | 4 | | | | II.4.b | |
| 3.05 | | B.4 | 4.a, 4.d, 4.g | 2.6 | | 8 | 4 | | | | 'II.4.e' | |
| 3.06 | | B.4 | 4.f | 2.6 | | 12 | 4 | | | | II.4.d | |
| 3.07 | C6, C9 | B.4 | l.c 3.d, 4.a | 1.3 2.5 | | 5,7, 9,22 | | | 2 | '2' | II.4.a | III.a |

| | AAES | ABET | ABET | ACM | ACM | BCS | BCS | ICCP | ECPD | IEEE | NSPE | PMI |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **ClntlEmp** | A.2 | | | | | | | | | | 'II.4' | |
| 4.01 | C.1 | B2, B5 | 2, 3.c, 5.b | 2.2 | 2.2 | | 1.3 | 2.5 | | 6 | I.2 | I.b |
| 4.02 | | | | 1.5 | | | | | | | II.1.b, II.2.c | III.a |
| 4.03 | | | 4.I.3 | 3.3 | 1.5, | 2 | | | | | III.7.c | III.a |

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | 5.o | | 1.7 | | | | | | | |
| 4.04 | | | | 1.5 | 1.5 | | | | | | | |
| 4.05 | C.5 | B4 | 4.I, 4.i. | 1.9, 11.8 | 1.9 | 2 | 4.5 | 2.1 | | | II.l.c, III.4 | |
| 4.06 | | | | 2.5 | | | | | | | | |
| 4.07 | | | 4.h | 1.5 | 2.5 | 26 | | | | | 'III.l.b' | III.a |
| 4.08 | | | | | | | | 3.6 | | | I.4 'III.1.c' | III.a |
| 4.09 | | | | | 2.6 | | | | | | I.4 | IIIa |

| | AAES | ABET | ABET | ACM | ACM | BCS | BCS | ICCP | ECPD | IEEE | NSPE | PMI |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Profession** | | | 4.I.4, 4.m | | | | | | 1 | | | |
| 5.01 | | | 4.b | | | | | | | | II l.d | |
| 5.02 | | A4, B6 | 6 | 4.1 | | | 13 | | | 10 | | I.e |
| 5.03 | C7, C10 | | 6.a | 4.1 | | | 13 | | 3 | 10 | | I.e |
| 5.04 | B, | B6 | | 3.1 | | | | | | | | I.e |
| 5.05 | C.9, | A4, B6 | | 4.2 | 1.2, 2.5 | | 13 | | 3 | | I.5, II.1.e | |
| 5.06 | | | 1.f, 6 | | 1.2 | | | | | 1 | II.1.a | I.a |
| 5.07 | | | 1.c, 1.d | | | | | | | | III.6.b | |
| 5.08 | C.3 | | 1.c. | 22.5 | | 10 | | 3.4 | 3 | | I.5, III.3.a | |
| 5.09 | | B.6 | 5.b | | | | | | 1 | | III.1.f | |
| 5.10 | A,B, C9, C10 | | | 2.3 | 2.3 | | | 3.2 | | | | I.b |
| 5.11 | | | 4 | 3.6 | | | | | | | 'III.2.a' | |
| 5.12 | | | | | 2.7 | 14 | | 2.2 | | | III.2.c | IV.b |
| 5.13 | C.7 | | 1 | | | 13 | 1.2 | 3.1 | 5 | | III.11a III. l1.b | |
| 5.14 | | | 3.a, 7.f | | | | 8,15 | | | 4 | II.5.b | |

| | AAES | ABET | ABET | ACM | ACM | BCS | BCS | ICCP | ECPD | IEEE | NSPE | PMI |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Colleague** | | | 7.c | | | | | | | 8 | | |
| 6.01 | C.7 | B.7 | 5.a | 3.6 | 3.6 | 18 | 1.2 | | 4 | '10' | III.2 'III.8.a' | II.g |
| 6.02 | | B.7 | | | | | | | | | | |
| 6.03 | C.8 | B.7 | 7.a | 1.6 | 1.6 | 16 | | | | 7 | III.10.a | II.f |
| 6.04 | | | 4.o | 16,19 | 2.6 | | | | | 7 | | |
| 6.05 | C.8 | | 5.i, 7.e | | | | | | 3 | | | II.c |
| 6.06 | | | 3.c, 5.m | | 4.1 | | 2.1 | 2.7 | | | III.11.e | II.c |
| 6.07 | | | | | | | 2.1, 4.5 | 2.7 | | | III.11.e | |
| 6.08 | | | | | | | 1.3 2.2 | | | | III.11.d | |
| 6.09 | | | | | | | | | 5 | | | II.c |
| 6.10 | | | | | | | | | | | III.10.c | |
| 6.11 | | | | | 16 | | | | | | III.11.e | |
| 6.12 | | | 4.k | 2.7 | | | | | | | III.11.e | |
| 6.13 | | | 5.d | | | | | | | | III.6.b | |
| 6.14 | | | 4.n, 7.i | | 16 | | | | | 8,9 | | |

| | AAES | ABET | ABET | ACM | ACM | BCS | BCS | ICCP | ECPD | IEEE | NSPE | PMI |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Self** | | | 7.g | | | | 1.1 | | | 6 | | |
| 7.01 | B, C7 | A3, B7 | | 2. | 2 | 18 | 1.1 | 3.1 | 4 | 6 | III.11a | I.c |
| 7.02 | B, C7 | A3, B7 | | 2.2 | | | 1.1 | 3.1 | 4 | 5 | III.11a | I.c |
| 7.03 | B, C7 | A3, B7 | 7 | 2.2 | | | 1.1 | 3.1 | 4 | | III.ll.a | I.c |
| 7.04 | B, C7 | A3, B7 | | 2.2 | | | 1.1 | 3.1 | 4 | 5.6 | III11.a | I.c |
| 7.05 | B, C7 | A3, B7 | | 2.3 | | 3 | 1.1 | 3.1 | | | III.11 | I.c |
| 7.06 | | A3, B7 | | 4.1 | 4.1 | | | | | 10 | | I.c |

*****TEMPORARY ADDRESS CHANGE****** Don Gotterbarn Centre for Computing and Social Responsibility DeMontfort University, School of Computing Sciences Leicester LE 1 9BH, UK

From: IN%"PRFCMP-L@UTKVMI.UTK.EDU" "Professional Competence Standards Task Fo r with spool id 5129 for PRFCMP-L@UTKVM1.UTK.EDU; Mon, 03 Feb 1997 09:17:10 +0000 Date: Mon, 03 Feb 1997 14:15:14 +0000 From: Donald Gotterbarn <dgot@DMU.AC.UK> Subject: Re: Code Annotation In-reply-to: <c=US%a= %p=EQT%1=ERIXCHANGE-9701301453512-8 Ed, As regards the schedule, I did draft a detailed schedule for completion of the project as was requested by the Steering committee. Until that schedule's details and the requested support are approved by them, I did not feel it was right to distribute it widely, but instead listed in general terms those steps which I hope will lead to a successful completion of our tasks.

I understand your concern and appreciate your enthusiasm, but I did not want to distribute anything which was not agreed upon.

regards, don

NOTES

[1] Gotterbarn Email (February 12, 2004) Attachment.

[2] Gotterbarn's computer arrangements were more complicated than this description suggest: "I had my laptop—which was a faster machine than the one supplied by DMU, so I brought it every day and did my work on it. I would then either transfer the data to the DMU machine and send it. Or I would go home in the evening and use my modem to connect and send messages." Gotterbarn Email (February 12, 2004), Attachment. Hence, Gotterbarn could send and receive email from home in the evening or on weekends.

[3] SEEPP's working groups had, of course, long since ceased to matter; anyone who did notice the reorganization was unlikely to care. Gotterbarn therefore had no reason to emphasize the reorganization. Compare Gotterbarn: "I formed a single committee consisting of the Prof Competence people and the working groups leaders from the other groups, hoping they would bring their functioning groups members along with them. PRF-Comp was just a mailing list, a tool that was there and that was useful. It existed so it was easy to add people to the mailing list. There were no hidden motives!! You make something of nothing. I did a complete reorganization. A Single purpose group for single –clear task assigned by steering committee. If I had a functioning mailing list labled 'happy day' I would have used it." Gotterbarn Chapter6cmt (October 31, 2004).

[4] See email from Gotterbarn to Mechler (February 3, 1997): "I did draft a detailed schedule for completion of the project as was requested by the Steering Committee. Until that schedule's details and the requested support are approved by them, I did not feel it was right to distribute it widely, but instead listed in general terms those steps which I hope will lead to successful completion of our task." This seems to me to be a change in the way Gotterbarn (and Gotterbarn-Melford) had operated until 1997. Gotterbarn-Melford had often announced schedules without Steering Committee approval. What explains the change? "By getting approval for a plan," Gotterbarn suggests, "I could proceed full speed ahead without asking permission at every turn." Comments on Chapter 6 (September 23, 2003). Why did Gotterbarn think he would otherwise need "permission at every turn"? The Steering Committee has already authorized him "to drive the task force to completion". The Steering Committee had also said he would distribute the plan "next week" but not that he needed anyone's permission to carry it out: "Other activities for the remainder of the year will be outlined in a plan that Don will distribute next week."

[5] The two references to Gotterbarn's own work are his entry, "Software Engineering Ethics", in the *Encyclopedia of Software Engineering* and one of the papers he gave at PASE'96, "Software Engineering: the new professionalism".

[6] Actually, Gotterbarn gives the publication date of Johnson's anthology as "1996" rather than "1990" (a typo or perhaps a misreading of a handwritten "0" as a "6"). The first edition of the Martin and Schinzinger text came out in 1983; the second edition (1989) was basically the same book. Both references were, then, substantially less current than the stated publication dates suggest.

[7] Comments on Chapter 6 (September 23, 2003). Gotterbarn\Gotterbarn memories of 1997 (9\1\2003).

[8] "I was not writing for scholarly consumption. For details on this [his scholarly views], see my 'The Moral Responsibility of Software Developers: Three Levels of Professional Software Engineering,' *The Jouranl of Information Ethics* 4 (Spring 1995): 54-64." Comment on Chapter 7 (September 21, 2005).

[9] Charles E. Harris, Michael S. Pritchard, and Michael J. Rabins, *Engineering Ethics: Concepts and Cases* (Wadsworth: Belmont, 1995), p. 38 (a code of ethics as the solution to a coordination problem, something more than just a guide to individuals).

[10] Was Gotterbarn aware that Mechler might object to the omission of standard-raising? Probably not, though he certainly had another opportunity to become aware of it. Four hours after sending off the statement, Gotterbarn got his George Washington email account to forward to his DeMontford address Mechler's response to Shaw. He did this, it seems, as part of a larger effort to collect all comments on the code before preparing his revision. That same evening he retrieved the comments of Frailey, Fulghum, Langford, and Prinzivalli in the same way. (Two days later, he also retrieved Shaw's original message.)

[11] There may have been an additional delay. At this time, my name had (again) dropped from Gotterbarn's list. Any email I received from Gotterbarn was a copy Weil passed on—after checking to see if I had received it independently.

[12] In his comments on this chapter (September 23, 2003), Gotterbarn raises a question of professional ethics: "Don't you wonder about the moral responsibilities of MD here?...Certainly he had responsibility as a professional to address that [that is, to correct what I considered Gotterbarn's omission of an important purpose of codes]." The question is worth answering here because it underscores the difference between the (1980s) theories of professional ethics Gotterbarn took for granted and the theory I developed in opposition to it (and which, apparently, he was not aware of even in 2003). My position is that I have no obligations "as a professional [as such]" , only as a member of a particular profession (or as a moral agent, parent, promisor, or other particular relationship). What is my profession? I'm an academic, a member of the professoriate. That's my only profession; philosophy is the discipline I practice, but not my profession (a special way to earn a living). To decide what my obligations as an academic are, I start with the code of ethics of the American Association of University Professors. I don't find any requirement there that an academic must correct every error anyone makes with respect to his discipline, not even that an academic should correct every error a colleague (or student) makes. There may, of course, be obligations the code overlooks (the unwritten ones that derive from the written or have some other origin). But, any claim for one of those requires an argument—and, in this case, I believe there is none. What Gotterbarn seems to have in mind is a general obligation to be a Good Samaritan. I think there is no such general professional

obligation. For more on this, see my *Ethics and the University* (Routledge: London, 1999), Chapter 4 ("Science: after such knowledge, what responsibility?").

[13] In 1997, ":-)" was still all that printed, though the meaning was a smile: ☺

[14] Comments on Chapter 6, September 23, 2003. Gotterbarn later added: "One of the reasons for the delay [in recognizing that the problem was in the kind of type the printer used] is no one of us considered that an ascii table would be printed using anything but a fixed font. We did not look for the problem here because we thought people knew better. When it was thought that this might be the issue, it was stated in a way not to embarrass anyone." Gotterbarn Chapter6cmt, October 31, 2004.

[15] For a decade or so (until the mid-1960s), the ECPD code had the endorsement of most American engineering societies. The NSPE's code is its direct descendent (and the ABET code, a radical reworking of it).

[16] Schinzinger was a professor of engineering already close to retirement when he teamed with Martin, a young philosopher, to do the first edition of their popular text. He might well have remembered the Faith from his youth and suggested its inclusion.

[17] February 3, 1997: "Most of the codes are in Ethics in Engineering, Martin and Schinzinger, and [sources for] the others are on the NET Reference Table Comparing Software Engineering Code v 1.0 to other Codes of Ethics, version 1.0 January 1997 Software Eng. C of E v1". "Most" is both a (gross) understatement and a (slight) overstatement. The six codes from Martin and Schinzinger are *all* the engineering codes but only half of the total (counting codes as the table does).

[18] Getting a copy of the code of ethics ("Ethics Policy") for British engineering, for example, for the Institute of Electrical Engineers, was then, and remains, difficult. Only someone teaching a course in engineering ethics in the UK would be likely to have one easily available. Codes of ethics for computing are much more accessible in the UK. My concern in noting what Gotterbarn did not do is not to say he should have done more (under the conditions), but to be clear about what he did do (and therefore about what he did not do), facts relevant to assessing the changes he made.

[19] Gotterbarn's Comments on Chapter 6 (September 23, 2003): "I guess we didn't do all those other things because we were paying attention to the TASK assigned- show that the draft code has roots in engineering and computing. Not law, medicine etc. The tasks as assigned were difficult enough to get done in the time frame without adding to the task." Exactly.

[20] This use of "committee" may be a mere "slip of the pen", but it may also be the first reference to the executive committee soon to be formally announced in the "Proposed Schedule". See Chapter 8.

[21] This, of course, is the only omission from Mechler's list of eight (originally sent December 19, 1996, but resent January 30, 1997). I am, of course, assuming here (for the reasons given in 6.9) that the December email never reached Gotterbarn.

[22] The question here is *not* who else contributed suggestions for revising the code. There were many (Frailey, Fulghum, Kanko's students, and so on, but not Mechler). Instead, the question is who provided comments or annotations concerning the "roots" of Version 1 in other codes.

[23] "Thanks for the credit—but Simon and Keith were always there." Gotterbarn Chapter6cmt, October 30, 2004. Yes, they were "there", Rogerson in person (when not out of town) and Miller by email. So, they certainly might have helped in constructing the table (hence, my hedging "more or less"). But no evidence survives of their participation in construction of the table—and neither of them actually recalls helping with it—though there is plenty of evidence of their helping on almost everything else. So, it seems to me, Gotterbarn really does deserve the credit I have given him ("more or less").

[24] Gotterbarn objects to this interpretation: "This is wrong—I do remember being greatly pleased by the work he provided. That is why I mentioned him specifically! GIVE HIM CREDIT! He was still working on the Code. YOUR THEORY MEANS I MUST HAVE LIED GIVING HIM SPECIAL THANKS." Gotterbarn Chapter6cmt (October 31, 2004). While I do not mean to suggest that Gotterbarn is here telling even a white lie, I do think his words give a misleading impression of what happened (however unintended). That Gotterbarn now remembers being please what Mechler's contribution is consistent with its merely providing a check on what Gotterbarn had already done (and with Mechler deserving the praise Gotterbarn gave him at the time). The timing seems entirely against Gotterbarn's claim that (in effect) five columns of the table are Mechler's work, not his.

[25] Gotterbarn\Version 1\Cmntster.

[26] Gotterbarn objects to my characterization of the executive committee: "It was a software engineering code—Miller an academic—does contracts for NASA etc., lots of experience on practical and academic side: same for Rogerson and Gotterbarn." Gotterbarn Chapter6cmt (October 31, 2004). True enough, but in addition to not being obvious to anyone who just looks at their institutional affiliations, these qualifications would, even if obvious, not satisfy an engineer that the perspective of engineers would be understood. Gotterbarn's criticism of this passage assumes that "software engineers", even if fully qualified to work on software, are necessarily engineers, an assumption most engineers would then (and even now) reject.

[27] Gotterbarn here misstates Version 1's structure. Five of Version 1's Rules (Public, Client and Employer, Profession, Colleagues, and perhaps Self) are "relationships" (in some sense) but the other two (Product and Judgment) certainly are not (or at least not in the same sense). Indeed, the long section on "product" is something unusual in codes. I had created it simply to hold a lot of suggestions not fitting any other category. I constructed Version 1 "from the bottom

up" (ignoring the relational or function structures of earlier codes); Gotterbarn was constructing Version 2 "from the top down" (or, at least, wanted the Steering Committee to think of the code in that way). Seeing what I had done, he assumed I had worked the same way. In fact, my intentions (or method) could not be read from the results achieve (the code itself).

[28] Gotterbarn's comments on the first draft of this chapter (September 23, 2003) respond to my question "Who but an academic would ask that?"—"An academic who had actually read various international Computing codes of ethics—like the Australian Computer Society's code which specifically mentions students." The academic who actually read the code in question is not a hypothetic academic, presumably, but Gotterbarn (and perhaps the other members of the EC). So, it is worth pointing out that, when I checked the ACS's code at http://www.acs.org.au/national/pospaper/acs131.htm (September 23, 2003), I found a code that mentioned "students" (here and there) but never "professors". Its section 4.3 (Values and Ideals) is typical: "I must act with professional responsibility and integrity in my dealings with the community and clients, employers, employees and students." The idea of specifically mentioning "professors" (as well as "students") may haave arisen from Gotterbarn's wide reading in codes of computing ethics and his wide experience with computer ethics more generally. What it did not arise from is any of the criticism he had so far received from anyone (or, at least, any that have survived on paper), even from criticism coming from professors and students—or from the code he actually cited.

[29] See, for example, Ronald Dworkin, *Taking Rights Seriously* (Harvard University Press: Cambridge, Massachusetts, 1977), esp. pp. 14-80; or the Walt Disney movie, *Pirates of the Caribbean* (2003), in which pirates unwilling to follow "The Pirate Code" when they should, say (something like), "Well, it's not so much a rule as a guideline."

[30] Gotterbarn's objection to "checklists" presupposes an answer to the prior question: *what is wrong with using a code of ethics as a checklist?* Using the code in that way certainly seems better than not using it at all (because, say, one does not know where to start and does not want to take the time to find out). Using the code as a (complete) checklist is a somewhat different matter. It too is better than nothing but definitely not as good as asking, after going through the checklist, whether there might be something the checklist should include but does not—or something that one should consider whether a code could include it or not. No doubt, there is something wrong with a question such as this: "Sexual harassment is prohibited, but what sort of harassment is not?"

[31] For a critique of some of these claims, with references, see my "Three Myths about Codes of Engineering Ethics", *IEEE Technology and Society Magazine* 20 (Fall 2001): 8-14 & 22, reprinted (with some revisions) in *Profession, Code, and Ethics* (Ashgate: Aldershot, England, 2002), pp. 121-132.

[32] Gotterbarn's memory of these events differs a bit from what my reconstruction: "When working hard many of us—ME—like to joke a lot. I inserted Simon's phrase ["outwith"] in several places just because of the reaction I knew it would draw from Keith. Simon did have a lot

to do with the Code, but this phrase was just a break from the work." Comments on Chapter 6 (September 23, 2003). So, according to Gotterbarn, it was his joke, not Rogerson's. This joke was the one piece of evidence I came across that anyone but Gotterbarn could change the text on his own, even as a joke.

[33] "The way email is processed in packets explains the other phenomena below. Again I tried to treat the frenzy caused by lack of knowledge about packets in a light hearted way. So I described the most extreme example and then suggested we move on and not further crowd the bandwidth." Comments on Chapter 6 (September 23, 2003)

[34] Actually, all addresses with "weil" were Vivian Weil's. I had a series of addresses with "csep" (or the capitalized version) before the "@", because I wanted CSEP's secretary to receive and print my email. My addresses were: csep@charlie.cns.iit.edu (1996-97); csep@charlie.acc.iit.edu (1995-96); and csep@IIT/VAX.BITNET (1994-95). The old addresses forwarded to the new addresses for a time, adding to the confusion, especially since IIT generally "migrated" addresses in batches, not all at once, and "csep", being near the beginning of the alphabet would be "migrated" well before "weil" near the end. What was happening at IIT was, of course, happening elsewhere as well. Within a month or two of the day on which Gotterbarn first put his list together, it was probably out of date and, no matter how hard he tried, it was probably never up to date again. Cutting-edge technology cuts both ways (making a job both easier and harder).

[35] Gotterbarn considers this email a "an explicit request for comment" (Comments on Chapter 7, September 21, 2005). I do not. The request is, of course, not addressed to me directly, but to Weil (since it was part of email attempted to get me back on the mailing list); it came after Version 2 was complete (and as part of attempting to send it out for generaly comment); and it lacked the specific questions Gotterbarn had addressed to members of his executive committee or DMU's inner circle. These considerations do not excuse me from commenting; they just address the question of whether was asked to comment (and on what).

[36] If Gotterbarn's transcription of my address is correct, the problem was simple but easy to miss, the confusion of "iit" (correct) with "itt" (incorrect). In general, email addresses require a perfection that postal addresses do not. In addition, during these years, email addresses changed much more often than postal addresses because they contained information about specific servers ("charlie" or "VAX"), administrative departments ("cns" or "acc"), and program ("BITNET") now generally omitted. Maintaining the address list turned out to be a permanent undertaking more demanding than keeping a paper list current. So, for example, something like the February 12 exchange occurred again on: March 27, 1997 (Gotterbarn to me, checking to see whether I got what he sent); April 7, 1997 (from Gotterbarn to me, "you have been added to the PRFCMP-L mailing list"—with detailed instructions); April 24, 1997 (from Gotterbarn asking for name, address, e-mail, and so on to "update database"); and April 29, 1997 (from me to Gotterbarn, thanking Gotterbarn for putting him back on the list but reporting that Weil, weil@charlie.cns.iit.edu, was now off).

[37] Gotterbarn never explains the switch from "Introduction" to "Preamble". I had chosen "Introduction" as a term less pompous than "Preamble" (and in keeping with Mechler's initial choice).

[38] The error is just one indication that Gotterbarn had prepared this memo in haste. Another is his description later in the paragraph of a preamble not yet written as if it were included in the document sent: "The code also differs from other codes by clearly addressing, in the preamble, the difference between the three levels of professional obligation and organizing the clauses under each principle around these three levels." The code never was so organized (and the preceding paragraph had admitted as much). Gotterbarn's thinking seems to have got a little ahead of his editing. That is not surprising given how much he did in his first five weeks in England. What is important, if not surprising, is how his thinking about the Code seems to have been evolving as theories espoused for years met the practical difficulties of writing a code.

[39] http://www.iit.edu/departments/csep/PublicWWW/codes/coe/Australia_Code.html.

[40] Gotterbarn gives no reference for this statistical claim. My random examination of a few computer ethics texts failed to confirm it. Gotterbarn's Comments on Chapter 6 (September 23, 2003) responded to this observation: "You claim that your readings fail to find any evidence for this claim is telling- Try reading the most influential book written on Computer Ethics- in the third edition now-'Computer Ethics' by Deborah Johnson- the point about professional relations is in all three editions." I then checked Deborah G. Johnson, *Computer Ethics* (Prentice-Hall: Edgewood Cliffs, New Jersey, 1985). What I found was a book with six chapters not much different from other computer ethics texts I checked. The chapter titles were: Ethical Theory; Professional Ethics; Liability for Malfunctions of Computer Programs; Computers and Privacy; Computers and Power; and Ownership of Computer Programs. The text was not organized by relationships. However, Johnson's chapter on professional ethics does contain a section, pp. 26-30, on " Professional Relationships" subdivided into four "relationships" (Employer-Employee; Client or Consumer-Professional; Professional-Professional Relationships; and Society-Professional). There is no claim there (or elsewhere in the book) that these relationships cover all the topics of professionalism or even that they are useful as a mnemonic. Johnson makes no pedagogical claim whatever there. So, it seems, Gotterbarn still lacks authority for his statistical claim. And, in any case, the software engineering code is, as explained above, only partly organized according to "relationships" (at least, relationships of the sort Johnson distinguishes).

[41] Or, at least, is true if we *ignore* the intended effect of the premising "In particular" clause. In many, perhaps most, instances, the conduct intended in Version 1 and Version 2 was probably much the same. The problem was finding the best words to achieve that effect (with only our knowledge of English to guide us). A "user lab" would have been a great help in preparing this code of ethics (or any other).

[42] "Level [or type] 1 obligations" were those that applied to humans as such (that is, moral obligations). Why they should be more, rather than less, aspirational is not stated. *Generally*, we consider moral obligations (the universal ones) to preempt merely professional ones. That

"generally" indicates some controversy concerning whether professional obligations may (occasionally) preempt moral ones. See, for example, two advocates of professional obligations as preempting ordinary morality: Benjamin Freedman, "A Meta-Ethics for Professional Morality," *Ethics* 89 (October 1978): 1-19; and Alan Goldman, *The Moral Foundations of Professional Ethics* (Rowman and Littlefield: Totowa, NJ, 1979). For some important criticism of this "separatist thesis", see: Paul R. Camenisch, "On Being a Professional, Morally Speaking," in *Moral Responsibility and the Professions*, eds. Bernard Baumrin and Benjamin Freedman (New York: Haven Publications, 1983), pp. 42-60; Alan Gewirth, "Professional Ethics: The Separatist Thesis," *Ethics* 96 (January 1986): 282-300; and Michael Davis, "The Moral Authority of a Professional Code", *Authority Revisited: NOMOS XXIX* (New York University Press: New York, 1987), pp. 302-337.

[43] October 6, 1996 (Mechler to Cabrera). Shaw had objected to the original 2.02: "This is too arbitrary. Why must their belief be well documented? Doesn't this depend on the intended use of the software? Doesn't 'safety' also depend on the intended use?" Mechler's response had been to repeat what he had said in response to a similar comment concerning 1.14: "Ethics is a set of principles or values that guide our judgments." He wanted to guide the judgment of software engineers toward using all appropriate tests, not just some, and to doing a good job of documenting their beliefs (and what they knew). What tests are appropriate and what is a good job of documenting may depend on intended use and is certainly a matter of judgment (among other things), but what Version 1 said, in effect, is: *Don't do without adequate documentation of your belief that the product is safe, meets specifications, and has passed all the tests you consider appropriate*. What objection could there be to that requirement?

[44] The principle is "questionable" here (and in interpretation of codes of ethics in general) but not everywhere. It is a standard principle in the criminal law ("No crime without a statute"). The principle is questionable in interpreting codes of ethics because a decent person does not try to do the minimum she can (without actually doing something wrong). The only way to be reasonably sure of not falling below the minimum is to resolve reasonable doubts in favor of the more demanding interpretation, not the less demanding. In ethics, there is something wrong with "sailing close to the wind".

[45] "[N]o—we put a much more limited definition on this." Comments on Chapter 6 (September 23, 2003). When I asked Gotterbarn what interpretation (and whether "this" meant "maintaining independent professional judgment", "conflict of interest", or 3.06), I got no answer. As the author, I would have like to know. Plainly, I had somehow failed. I would have liked to know how. But time has probably closed the door to such knowledge.

[46] The conclusion to draw here is that it would be good to have the knowledge that Gotterbarn, Miller, and Rogerson claimed but did not have and good too to admit that they did not have it.

[47] The (British) "whilst" suggests that Rogerson wrote this comment. Gotterbarn, however, disagrees: "Simon did have a lot to do with the Code, but his phrase ["outwith"] was just a break

from the work. Just as My use of the word whilst—whilst I am in England….A hunch—but Don may be funny here because the OED was given as the Source rather than Webster. I actually vaguely recall Duncan [Langford] getting into this issue." Comments on Chapter 6 (September 23, 2003). The crucial email from Langford, if there was any ("vaguely recall"), seems not to have survived.

[48] See, for example, *Webster's New Twentieth Century Dictionary of the English Language, 2d ed*. (World Publishing Company: Cleveland, 1960), p. 606; or *Thorndike-Barnhart Comprehensive Desk Dictionary* I (Doubleday & Company: Garden City, New York, 1958), p. 275. If these references seem too old, going to the current Merriam Webster website yields much the same: "to make sure, certain, or safe**: GUARANTEE**. **synonyms** ENSURE, INSURE, ASSURE, SECURE mean to make a thing or person sure. ENSURE, INSURE, and ASSURE are interchangeable in many contexts where they indicate the making certain or inevitable of an outcome, but INSURE sometimes stresses the taking of necessary measures beforehand, and ASSURE distinctively implies the removal of doubt and suspense from a person's mind. SECURE implies action taken to guard against attack or loss."

[49] For example, while the OED's first non-obsolete definition of "assure" is "to make safe *against* or *from* (*of* obs.) risks; to insure, *esp.* on mod. usage **to assure life**: to secure the payment if a specified sum in the event of death", the first non-obsolete definition of "ensure" is: "to pledge one's credit to (a person); to tell (a person) confidently *that* (something is true)."

[50] Of course, "compensation" carries this meaning in the US as well; the question is whether "Europeans" (or, at least, the British) would also understand that the sense intended is not payment for wrong but payment for work. A visit to the OED reveals that while "compensation" in the sense of salary is an Americanism (2d), the British do use "compensation" (2a) to mean, "That which is given in recompense, an equivalent rendered, remuneration, amends." So, if OED offers a good description of British usage, and even if Europeans still learned British English rather than American English, "compensation" should have caused "the Europeans" no trouble.

[51] Gotterbarn\Version1\CMTFULG. I had considered having a similar section early in the preparation of Version 1 (with the keyword "Supervisors") but had given up because there were then too few clauses specifically dealing with supervision. Instead, I had treated supervision as just one of the activities of "colleagues". Once I had the seven principles, I did not notice that the suggestions coming in had swelled the clauses under Principle 6 to the point that division into two principles, one concerned with colleagues strictly so called and one concerned with software engineers working as supervisors, had become not only possible but desirable. Not then having the benefit of Fulghum's suggestion, I had missed an opportunity to make the code substantially more "user-friendly"—as did the rest of SEEPP/E.

[52] Version 2 was, of course, in part a response to outside comment, but that is not the same as having the new version put to the test of outside comment. No code writer knows whether a provision will pass muster until it has. Every change in a code, even the change of punctuation, is

a bet that others will see in it what the author does—or, at least, not see anything to object to. Just as the authors of Version 1 lost many of their bets, so the authors of Version 2 would lose many of theirs.

[53] Gotterbarn asks: "Why late in the game—this was a draft that had not passed muster from the community it would serve?" Comments on Chapter 6 (September 23, 2003). We have here a question about how to interpret the set of facts we have before us (Chapter 6). I say "late in the game" because (technically) "the game" was already in over time. SEEPP had already missed the original deadline for completion of the code, and was now working beyond the second deadline. (That the "game" in fact went on another three years does not change these facts—or the perception they should have produced at the time.) *My* reading of the comments of the community the code was to serve had been quite positive, except for those of Failey and Shaw. And, on the whole, the criticism of Frailey and Shaw seems to be more a matter of misreading the code, or simple ignorance of other codes, than hostility to what the code actually said. What constitutes "passing muster" here is a political question, not a factual one. Gotterbarn may well be right about what was necessary to get the Joint Steering Committee to pass the code, any code. But I have no idea how to decide that he was right. And I am pretty sure he doesn't either. So, the best I can do here is set out the alternative interpretations.

[54] Gotterbarn thinks this explanation of his conduct in the preceding two years to be "nonsense". In particular, he pointed out in comments on an earlier draft of this chapter: "What was going on was not the engineering method—it had nothing to do with large projects. I have managed long-term multi-person projects. The problem was the documentation and review method and the way it was being given lip service." Gotterbarn Chapter6cmt (October 31, 2004). Gotterbarn now has no more access to his thinking in late 1994 than I do. We are both involved in reconstructing it from the evidence now available. One problem with his reconstruction is that it leaves unexplained why he initially went along with IEEE approach instead of insisting on the ACM approach or working out a compromise, and why he later changed, exactly what my reconstruction does explain (and what it set out to explain). Gotterbarn may be right that my reconstruction is wrong but, as far as I can tell, the evidence makes my reconstruction the more plausible. IEEE procedures got a good deal more than "lip service" 1994-96. The CFP, Operations Guide, and SEEPP organization are "body service" (deeds) rather than "lip service" (mere talk).

[55] Gotterbarn/94-95 misc/Boston 6-94.

[56] See http://whatis.techtarget.com/definition/0,289893,sid9_gci214112,00.html (September 4, 2004): "Typically, a skunkworks has a small number of members in order to reduce communications overhead. A skunkworks is sometimes used to spearhead a product design that thereafter will be developed according to the usual process. A skunkworks project may be secret."

# Chapter 8: English Spring, Version 2a-2.1

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

## 8.1 A schedule proposed

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

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

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

practicing software engineers and ethics advisors for major, multinational corporations. The panel also has an international representation from places such as Egypt, England, Germany, and Scotland.

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

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

Earlier (Chapter 7) we wondered what point there could be to formalizing Gotterbarn's "kitchen cabinet" as an "executive committee" and the remnant of SEEPP volunteers as a "panel of experts". We may now have at least a partial answer. Each of Gotterbarn's "deliverables" comes with a brief description of process. For example, under *a* ("the Schedule"), Gotterbarn reports: "After review by SEEPP executive committee, Delivered to Chairs of the Steering Committee". The same phrasing occurs under *b* (Statement of Architecture). The phrasing under *c* (Statement of Relations of Code to other Codes) is different: "After review by SEEPP panel of experts, delivered as appendix to Code Architecture to Chairs of the steering committee". The phrasing under *d* (the Code of Ethics) implicitly combines these (and perhaps other procedures):

"After review by several entities, redrafted and delivered to chairs of steering Committee". The small group that we had so far observed work informally and in great haste now sounds like a vast organization following its own leisurely procedures. That effect was achieved at the cost of a few titles and some careful phrasing (and without any departure from the truth).

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

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

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

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

Gotterbarn then laid out the "very tight" schedule. The first date (after February 14) was February 28 by which time the *Steering Committee* was to:

> Return Code v2 with comments to the task force,
> Return comments on any other documents
> Indicate commitment to schedule including
> - booking space in society publications for publication of draft code and ballot/survey in September Issues
> - support of WEB site at CCSR
> - support of announcement in society publication of existence of site and invitation to comment
> - permission to publicize this schedule to task force

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

> 1 March
>
> Task Force begins addressing steering committee concerns and continues revision of the code.
>
> Establish a web site at Centre for Computing and Social Responsibility on behalf of the ACM and IEEE-CS for development and dissemination of the Code. CCSR has significant international email lists and it is where I am during the development of the code. The web site could be managed by them and then turned over to the societies at the end of the project.
>
> 1 April
>
> Revised Code and working papers to Steering Committee
>
> 15 April
>
> Place draft copy of Code on web site for general comment.
>
> 1 June
>
> Completion of Code and associated documents for review by Steering Committee. Begin preparation of survey instruments for Code
>
> 15 June
>
> Revisit documents in light of Steering Committee comments
>
> 30 June
>
> Finish Revisions

8.2 Fairweather and Prior

Though the Schedule seemed to call for an immediate response, the Steering Committee did not respond immediately. Indeed, it would not respond for more than a month (and then not as Gotterbarn had planned). Gotterbarn could have waited impatiently, as he had in earlier years. Instead, he pushed on, simultaneously writing the Preamble and revising the code. He seldom

worked out anything alone. Sometimes he would first talk over his ideas with Rogerson. Sometimes, he and Rogerson would sit together at the same computer screen, discussing comments as they came in. Once Gotterbarn thought he understood what was needed, he would prepare a draft and circulate it to Rogerson, Miller, and two others at CCSR whom Gotterbarn had come to respect, Ben Fairweather and Mary Prior.

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

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

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

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

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

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

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

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

## 8.3 Preamble trouble

On March 18, 1997, less than two weeks before the revised code was to go to the Steering Committee, Gotterbarn emailed the listserv the promised preamble (Preamble v 2.0). What was a single paragraph in Version 1 had become eight paragraphs. The Introduction (name changed to "Preamble") was now a little essay, a page and a half long, single spaced. The first paragraph resembles the original. The first sentence is identical except for the addition of "education" and "social affairs" to the list of roles (previously "commerce, industry, government,

medicine, entertainment, and ordinary life). The second sentence replaces the original (partial) definition of "software engineer" ("those who design, develop, and test software") with another ("[those] who contribute, by direct participation or by teaching,[24] to the design and development of software systems").[25] The chief change in the third sentence is the addition of "must". Instead of describing software engineers as people who "commit themselves to making the design, development, and testing of software a distinct, beneficial, and respected profession", it now demanded: "[they] must commit themselves to making the design and development of software a distinct, beneficial, and respected profession." Apparently, one can be a software engineer without such a commitment (the "must" of command always implying the alternative as a possibility).[26] The fourth sentence is exactly the same in both versions ("In accordance with that commitment, software engineers shall adhere to the following code of ethics").[27] But the last two sentences of Version 1, the ones that Frailey and Shaw had criticized[28], are gone, their place taken by seven paragraphs each about the same length as the first.

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

The next four paragraphs offer advice on how to use the code. The first of these—the fifth of the eight—warns that the code is "not intended to be all inclusive nor is it intended that its individual parts should be used in isolation to justify errors of omission or commission"; it is also "not exhaustive" and not intended to be "read as separating the acceptable from the unacceptable in professional conduct in all practical situations." The code is not a "simple ethical algorithm". The fifth paragraph having explained what the code is not, the sixth explains what the code is, "[an attempt to] influence you to consider broadly who is affected by your work; to examine if we are treating other human beings with due respect;….and to consider if your acts would be considered worth[y] of the ideal software engineer." Software engineers should

interpret the code in such a way that they can answer yes to the question, "[Have I acted] in a manner likely to be judged as the most ethical way to act in circumstances by informed, respected, and experienced peers in possession of all the facts"? The seventh and eighth paragraphs explain the utility of the code. The code "provides support for the software engineer who needs to take positive action by documenting the ethical stance of the profession". It also has an educational use, "stating what is required of anyone wishing to join or continue in the software engineering community". In sum, "[because] it expresses the consensus of the profession on ethical issues[,] it can be used as a guide to decision making and as means to educate both the public and aspiring professionals about the professional obligations of software engineers."

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

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

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

## 8.4 The problem of silence

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

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

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

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

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

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

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

Many similar changes, large and small, are scattered through Version 2.0a. Among the most important is a series suggesting that Gotterbarn thought the process of weakening the code (making it "more aspirational") may have gone too far. The first hint of this is in the Preamble. The last sentence of the sixth paragraph now read: "Since this code represents a consensus of those engaged in the profession[,] one should TAKE INTO ACCOUNT WHAT IS likely to be judged as the most ethical way to act in the circumstances by informed, respected, and experienced peers in possession of all the facts AND ONLY DEPART FROM SUCH A

COURSE FOR PROFOUND REASONS, BACKED WITH CAREFUL JUDGEMENT".[37] Though not an algorithm, the code is to be more than a "guideline". Software engineers are not free to interpret it any way they feel is right. They are to take account of what "the profession" would do—operationalized as the probable judgment of one's "informed, respected, and experienced peers". To depart from the code so interpreted, a software engineer must have "profound reasons", nothing less.[38] There will, then, seldom be reason to depart from the code (properly interpreted). In this respect at least, Version 2.0a lays down rules strictly so called (though rules for which there may be a few exceptions). Though the Preamble's fifth paragraph contains new language that may seem designed to weaken the code—the insertion in its last sentence of "spirit of the" between "consistent with the" and "code"—, it is in fact part of a larger revision having the opposite effect. The last sentence of the sixth paragraph provides an interpretation of "in the spirit of the code" (indeed, a relatively demanding principle for interpreting the code). What is in the spirit of the code is generally acting as the most informed, respected, and experienced members of the profession would interpret the code, departing from that standard only for profound reasons.

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

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

Version 2.0a was still plainly a work in progress. So, for example, the switch from "assure" to "ensure" remained incomplete. Principle 1 itself retained "assure", though "assure" no longer appeared in any of the clauses under it. The first two of the nine clauses of Principle 5 also retained "assure" (as they did when they had an N in front of them in Version 2.0). 5.01 read "Assure that employees are informed of standards before being held to them" and 5.02 "Assure

employees know the employer's policies and procedures for protecting passwords, files, and other confidential information." "Assure" also appeared in 4.02 ("Assure that any document upon which they rely has been approved by someone authorized to approve it"). Since these clauses were the same as in the previous version, we might suppose that they remained the same because Gotterbarn had not looked closely at them, his attention focused on the clauses he was rewriting. But that explanation will not do. Clause 6.02 retained "assure", though revised to read: "Assure that clients, employers, and supervisors know of the software engineer's commitment to this code of ethics, AND THEIR OWN RESPONSIBILITIES UNDER IT."[40]

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

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

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

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

Of the remainder of the list, at least two (Fairweather and Prior) had commented on the code but never asked to be on the task force. Four others (Fulghum, Kanko, Langford, and Prinzivalli) had both joined the task force in response to the Call and had commented on one draft or another. Jewett, Kallman, and Phillips had joined the task force in response to the Call

but had not commented on the code. Of the several dozen who had joined the task force in its first year and did nothingafterward, they are the only three listed.[43] Peter Barnes (British) seems neither to have joined the task force at any time nor to have had anything to do with the code. His presence on the list is a mystery (not least to him).[44] John Weckert (an Australian), whom Gotterbarn had recently seen at a conference, had (again) expressed an interest in the project but had not yet done anything. [45] Missing from the list—beside Weil and me—is the German "expert" (or, indeed, anyone from the Continent) that Gotterbarn had claimed for the task force ("the panel of experts") less than two months before.

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

## 8.5 The Steering Committee responds

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

That was the Committee's first response to Gotterbarn's Status Report. It was, in a back-handed way, good news. The Steering Committee had broken its long silence and, in effect, accepted the schedule Gotterbarn had sent them in February and tried to keep to ever since. Gotterbarn's "Status Report" had worked. The Steering Committee had acted with unbelievable speed (meeting, it turned out, by conference call).[48] Frailey's email itself seemed to show that the

Committee understood the need for speed (or, at least, feared its message might be lost). The email began, "Don, please acknowledge receipt of this." Unfortunately, that was the only evidence that the Committee appreciated Gotterbarn's worries. The Committee had not done what Gotterbarn had asked (comment on the code, approve establishing a web site, and help with getting the code published in appropriate journals). Instead, it had done precisely what he had asked them not to do; it had introduced "small delays" (the time necessary to get their approval of the cover memo) that could "cause the project to drag out".

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

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

Gotterbarn read both emails when he returned to Leicester Wednesday evening, but waited till the next day (April 10) to respond. There was much to think about—not least those listed as attending the meeting: Cabrera was there only briefly and, apparently, only read "some thoughts" of the absent Zweben (*ex officio* in virtue of being ACM President) had asked him to give the Committee. Boehm and Shaw (both ACM-appointed members) were also absent. Even meeting by conference call, the Committee had in fact had only six of its ten members present for most of the meeting (only two of them ACM appointees). Was the Committee, or at least the ACM-appointees, losing interest? In spite of the low attendance at the meeting, the Committee agreed to meet "regularly" by conference call, "perhaps every three weeks" (at a time "to be agreed on by committee members"). Had the Steering Committee failed to respond to Gotterbarn over the last two months because it had not been able to arrange a face-to-face meeting? There was evidence of that. Though apparently called in response to Gotterbarn's Status Report, a third

of the meeting (or at least a third of the minutes) was devoted to the "Survey for Body of Knowledge". The body of knowledge task force had completed its survey (or, rather, a pilot) and was to have certificates just as the SEEPP task force was. "All" members of the Committee were to "respond to the call for dialog [concerning the survey] with well-conceived analysis and recommendations—Deadline will be April 30, 1997." There was a similar "action" for the code of ethics. "All" were to provide "feedback to Don Gotterbarn on the draft code by no later than May 31." There was nothing about the curriculum task force.[53] Was it still waiting for the body of knowledge task force to complete its work?

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

## 8.6 Pressing forward

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

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

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

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

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

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

This should have satisfied Feldman. Gotterbarn had almost done just what the Steering Committee had asked. He had written the cover memo, though in the form of an essay from his executive committee, not a memo from the Steering Committee. Gotterbarn had suggested that Feldman should find no fault with the first two paragraphs of the essay because, after all, they were only an abstract of an earlier article, one Barbacci (the first chair of the Steering Committee and now an ex officio member because he was IEEE-CS president) had already approved. The other three paragraphs in the essay were, while unconnected with Barbacci, basically true, (more

or less) necessary, and pretty much what the Steering Committee seemed to want (a statement of what the draft "is and is not"). The code was a draft distributed "to solicit comments"; the Steering Committee had reviewed the draft; and so on. Yet Feldman did not immediately respond, "Okay, I'll clear this with the Committee as fast as I can." The "cover memo" was now a roadblock Gotterbarn had to get around—quickly.

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

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

Gotterbarn's email to Barbacci would seem as cool as the email to Feldman were it not for its closing paragraph: "I really appreciate the promptness of your responses. It is fun to work on a project that is moving along. Thanks." There followed a "ps": "I will get the addresses, email, and fax information for the executive committee awardees to you shortly." Why Gotterbarn did not email Chikofsky directly is not clear. Gotterbarn seemed to be treating Barbacci as if he were still the Steering Committee's chair. In any case, that is the last we hear of the awards in the emails, though Gotterbarn did take time on Friday to straighten out some email addresses. He had tried to make these changes on April 7 (before sending off the status report), working through his George Washington University account. Apparently, that attempt had failed.

The very next day he had received a short response to the status report from someone whom he had just removed from the list.[60] He now wrote to the person in charge of listservs at ETSU, asking that seven addresses be removed (including one mistakenly assigned to me) and that four others be added (my correct address, as well as addresses for Barnes, Jewett, and Weckert).[61] SEEPP no longer consisted of everyone who applied (and did not resign).

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

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

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

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

Gotterbarn's files contain an email from Rogerson that seems to explain why Gotterbarn did not accept Feldman's paragraph. The email includes a "snip" from Miller. Preceding the full text of Feldman's paragraph, Miller had written: "I agree [apparently to some comment lost by

the snipping]. This makes our code sound even wimpier than it is. I suppose we must bow to the pressures and be somewhat spineless. But we don't have to BRAG about it, for goodness sake." (What seems to have bothered Miller is that Feldman describes the Code as "ideal behavior".) Miller then offered this "reword" of Feldman's paragraph:

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

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

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

### 8.7 Europe responds to Version 2.1

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

After expressing appreciation for the "the effort of the IEEE [sic] task force", it listed nine "general observations…made by the group", seven specific "remarks" (organized by Principle), and five "Concluding remarks". Some of the general observations sound much like the comments Shaw or Frailey made on Version 1: "the code needs to have its items prioritized in a more evident manner. Items need 'stacking'"; "there are conflicts among some of the items included in the code"; "the code represents idealism; then there is realism, the real world"; "many of the issues raised are not specific to software engineering"; and "the code is really a set of guidance notes". Other general observations seem to respond specifically to Version 2.1's innovative Principle 5 ("Management"): "the role of the individual as distinct from the role of the organization should be more clearly delineated"; "would some of the issues be more easily incorporated under a code of business ethics?"; and "are some of the issues raised actually about the management of large projects?" One of the general observations asks about "enforcement" (a question Feldman's paragraph might have averted). The final observation seems to treat as a

vice one of the virtues of the IEEE-CS/ACM effort, its attempt to involve as many people as possible: "some of the names included in the task force appear to be relatively 'new' names to the field of ethics". (The IFIP does not say which names appear new, but certainly the new names must include Burnstein, Mechler, Prinziavalli, and Fulghum, each of whom had done more than some of the well-known names.)

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

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

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

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

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

## 8.8 American responses

On May 22, Walter Maner, a professor of computer science at (Ohio's) Bowling Green State University, sent a brief email. After congratulating Gotterbarn on the code's "organization and content", he pointed to what he thought was "one large omission". Those software engineers that do "human factors" research often use human subjects. The code should have a section on human subjects research much like that codes of medical ethics have. There should be something about "informed consent, rights of the human subject, responsibilities of the experimenter, and confidentiality." Maner volunteered to "take a crack at drafting a set of guidelines". The offer

was never taken up. Gotterbarn does not recall why but suspects it just got lost in the rush of events. Ordinarily, he would have jumped at the chance to work with Maner.[68]

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

The student nonetheless concedes that a number of rules under Principle 5 address the issue "somewhat". 5.01 tells management to "assure that employees are informed of standards before being held to them" and 5.02 that management "assure employees know the employer's policies and procedures for protecting passwords, files, and other confidential information". But he would like to the code to state "a more explicit policy towards email in the work place." Having made his complaints, he ends his email by listing what he likes "best" in the code. All three items on his list date from Version 1: First, he liked the "last set of principles" because "a software engineer needs to maintain his competence" and "education does not end with a degree". Second he liked Principle 6, "about a software engineer's responsibilities to his profession", especially 6.13 ("Share useful software-related knowledge, inventions, or discoveries with the profession, for example, by presenting papers at professional meetings, by publishing articles in the technical press, and by serving on the profession's standard-setting bodies") and 6.04 ("Help develop an organizational environment favorable to acting ethically"). Last, he was glad to see "concepts" included in Version 2.1 that were missing from the ACM code, in particular "an emphasis on documentation and testing of software", "having qualified people work on a project", "warning the public about the dangers and risks associated with computers", and "[not asking] employees to do something inconsistent with the Software Engineering Ethics Code".

Within a week, two more emails arrived from Eau Claire. On May 26, "Student 2" sent a half page of comments. His are much more general than Student 1's. The introductory essay says that "v 2.1" is based on several other computing and engineering codes. He would like to have the source of each particular provision somehow identified within the code (perhaps as annotations). (No doubt, he would have been happy with something like the table Gotterbarn sent the Steering Committee in February.) Student 2 would also like to know what moral theory the

codes in question were based on, for example, “were [they] based on Kantian or Utilitarian principles”? The Preamble mentions three levels of ethical obligation but “there is a deficit of pragmatic examples that could potentially identify and illustrate [the] ethical obligations.” He would like to see some “real life computer related examples” to help the reader “identify with them”. Last, Student 2 wondered whether the list of keywords might be “too extensive in number”. Perhaps some of “the ideas would best be presented in paragraph format with the intent of enticing the readers…[to] process this information.”

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

## 8.9 Mechler checks in

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

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

> Sorry to have been slow about this. The Draft code is out for comment in several domains.
>
> It is on the ACM web site www.acm.org in the serving the community area.
>
> It is, as Ed indicated, on the IEEE web site, but somewhat hard to find.
>
> It has been published in the June issue of the SIGCAS Newsletter and is coming out in the SIGSOFT newsletter.

> It should also be out soon in the TCSE-IEEE newsletter. I believe this is the largest IEEE-Computer Society technical group.
>
> All of these presentations ask for comments. There have been no responses yet. We will organize the comments when they start to come in and present them to the task force for discussion and vote. We have not had any new comments from any steering committee member.
>
> Not much else new.

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

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

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

8. Appendix:

**Draft Software Engineering Code of Ethics**

*Don Gotterbarn, Keith Miller, Simon Rogerson*

version 2.1

April 1997

---

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

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

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

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

The draft evolved after extensive study of several computing and engineering codes. All these codes try to educate and inspire the members of the professional group that adopts the code. The code also informs the public about the responsibilities that are important to a profession. Codes instruct practitioners about the standards that society expects them to meet, and what their peers strive for and expect of each other. Codes are not meant to encourage

litigation, and they are not legislation; but they do offer practical advice about issues that matter to professionals and their clients and they do inform policy makers. Based on the feedback from readers of this publication and from other sources, a final draft of the code will be developed and presented to the Steering Committee for approval.

---

**PREAMBLE v 2.1**

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

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

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

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

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

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

situations require the software engineer to use ethical judgement to act in a manner which is most consistent with the spirit of the code of ethics, given the circumstances.

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

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

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

---

**PRINCIPLES v 2.1**

**Principle 1: PRODUCT**

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

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

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

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

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

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

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

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

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

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

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

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

1.12 Delete, whenever appropriate, outdated or flawed data.

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

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

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

---

**Principle 2: PUBLIC**

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

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

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

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

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

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

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

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

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

2.09. Accept full responsibility for their own work.

---

**Principle 3: JUDGMENT**

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

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

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

3.03. Reject bribery.

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

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

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

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

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

---

**Principle 4: CLIENT AND EMPLOYER**

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

4.01. Provide service only in areas of their competence.

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

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

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

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

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

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

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

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

---

**Principle 5: MANAGEMENT**

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

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

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

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

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

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

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

5.07. Offer only fair and just remuneration.

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

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

---

**Principle 6: PROFESSION**

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

6.01. Associate only with reputable businesses and organizations.

6.02. Assure that clients, employers, and supervisors know of the software engineer's commitment to this code of ethics, and their own responsibility under it.
6.03. Support those who similarly do as this code requires.
6.04. Help develop an organizational environment favorable to acting ethically.
6.05. Report anything reasonably believed to be a violation of this code to appropriate authorities.
6.06. Take responsibility for detecting, correcting, and reporting errors in software and associated documents on which they work.
6.07. Only accept remuneration appropriate to professional qualifications or experience.
6.08. Be accurate in stating the characteristics of software on which they work, avoiding not only false claims but claims that might reasonably be supposed to be deceptive, misleading, or doubtful.
6.09. Not promote their own interest at the expense of the profession.
6.10. Obey all laws governing their work, insofar as such obedience is consistent with the public health, safety, and welfare.
6.11. Exercise professional responsibility to society by constructively serving in civic affairs.
6.12. Promote public knowledge of software engineering.
6.13. Share useful software-related knowledge, inventions, or discoveries with the profession, for example, by presenting papers at professional meetings, by publishing articles in the technical press, and by serving on the profession's standard-setting bodies.

---

**Principle 7: COLLEAGUES**

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

7.01. Assist colleagues in professional development.
7.02. Review the work of other software engineers, which is not in the public domain, only with their prior knowledge, provided this is consistent with safety.
7.03. Credit fully the work of others.
7.04. Review the work of others in an objective, candid, and properly-documented way.
7.05. Give a fair hearing to the opinion, concern, or complaint of a colleague.
7.06. Assist colleagues in being fully aware of current standard work practices including policies and procedures for protecting passwords, files, security measures in general, and other confidential information.
7.07. Not interfere in the professional career progression of any colleague.
7.08. Not take steps to supplant another software engineer after steps have been taken for employment.
7.09. In situations outside of their own areas of competence, call upon the opinions of other professionals who have competence in that area.

---

**Principle 8: SELF**

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

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

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

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

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

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

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

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

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

---

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

**Chair: Donald Gotterbarn**

**Executive Committee: Keith Miller and Simon Rogerson**

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

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

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

NOTES

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

[22] Email, March 17, 1997.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

[44] Email (Barnes to Davis), January 13, 2003: "I have no recollection of ever being involved in any way in this process. In view of the fact you incorrectly mailed Survive Ltd as "Survival" I wonder if there may be another Peter Barnes in such an organization out there somewhere? Alternatively it may be that I spoke to someone about the development of a code of practice for business continuity and disaster recovery planning professionals while I was at Survive. If so, I

have no specific recollection and doubt that I would be able to contribute in any way to your current research." Gotterbarn (who provided the email address) thinks this is the right Peter Barnes: "[He] may not remember his involvement. He asked to be involved early on via email—his son responded saying Peter Barnes would be interested. We put Peter Barnes on the list. As I recall, I don't think Peter Barnes ever responded to any further emails about the Code. Five years ago…he may not remember any of it." Whatever Barnes's contribution, it must have occurred very quickly. Gotterbarn added Barnes to the listserv not "early on" (as in 1994 or 1995) but on April 11, 1997 (after trying, and failing, to do it on April 7)—and that is the last we hear of him (except for his inclusion in the list of Version 2.1's "authors")—or rather, the last we hear of him until Gotterbarn and Rogerson deliberate about including him among the "authors" of Version 3 even though he contributed nothing to it and gave no indication of wanting his name listed. His name remained on all subsequent versions. Gotterbarn\Version 2-2a-2.1\Task force vote\SR1. See Chapter 8 for more details.

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

[46] See Chapter 9 for the details.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

[64] Gotterbarn\version2-2a-2.1\delayed IFIP. A SIG is a subdivision of a Working Group (here WG 9.2), itself a subdivision of a Technical Committee (here TC 9).The author of the file is Jacques Berleur (whose work on codes of ethics Gotterbarn had cited in his February background paper on codes); the computer date on the file is October 7, 1997, suggesting that

Gotterbarn may not have received the document until then. This is consistent with the minutes' concluding paragraph: "Jacques Berleur asked members of the SIG to send him any further comments on the code. He offered to be in contact with Don Gotterbarn and forward to him our observations." Apparently, not realizing how fast Gotterbarn was working, Berleur took several months to collect comments before writing up the minutes. By then, Berleur was a veteran of the recently failed attempt of the IFIP to enact its own code of ethics. For very interesting details, see Jacques Berleur and Marie d'Udekem-Geves, "Codes of Ethics: Conduct for Computer Societies: The Experience of IFIP", in *Technology and Ethics: A European Quest for Responsible Engineering*, Philippe Goujon and Bertrand Hériard Dubreuil, eds. (Leuven, Belgium: Peeters, 2000), pp.327-350.

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

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

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

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

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

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

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

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

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

# 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)

# Chapter 10: Slogging Toward "Version 4.DONE"

"Build a system that even a fool can use and only a fool will want to use it."
—Murphy's Technology Law # 49

## 10.1 Another worrying silence

About the time Version 3.0 (9. Appendix) appeared in the October 1997 issue of *Computer*, Gotterbarn began to receive comments from (more or less) ordinary members of the IEEE-CS and ACM. The first arrived on October 11.[1] It was from Stephen Unger. Then chair of Columbia's Department of Computer Engineering, Unger had long been prominent in engineering ethics.[2] In part, his prominence was scholarly. He was the author of *Controlling Technology: Ethics and the Responsible Engineer* (1994) as well as of several much-discussed articles.[3] In part, though, his prominence was practical. Long active in IEEE, Unger had an important part in drafting three codes of ethics. The first, the IEEE code of 1979, had been replaced (in 1987) by the second he worked on. That one survived only three years, its short life a record in the history of codes.[4] What replaced it was the much shorter (ten-commandment-style) code still in force in 1997 (and today). Unger had no part in writing that one. Between these codes, he had helped to write one more, the innovative but soon-forgotten 1984 code of the American Association of Engineering Societies (AAES). Unger had bad luck with codes.

Unger found Version 3 "an impressive document, full of interesting, important points, covering a wide range of topics in considerable detail." Having noted that Gotterbarn's committee had "clearly decided to go for completeness as opposed to brevity, the opposite of the choice made by the IEEE, he expressed some satisfaction (in parentheses) that "the IEEE Ethics Committee plans to attempt to get the best of both approaches, by developing, over a period of time, an extended set of guidelines for the [1990] IEEE code."[5] Then, after apologizing for not giving the Code the "careful study that it deserves", he made ten specific suggestions that together give the impression that he must have read the code carefully. (He also referred Gotterbarn to the 1991 Swedish Ethical Rules for Computer Professionals that he had appended.) Unger's first suggestion gives a good indication of the rest: "Although your item 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'] is similar, the wording of item 3 of the Swedish code is very nice and perhaps might suggest some adjustment in your statement…: 'only take part in projects with the time and resources to do a good job.'" (Unger's codes tended to sound as if Hemmingway, rather than a corporate attorney, had done the writing—they were always easy to read.) Though Version 4 did not incorporate this suggestion, it did incorporate several others, including Unger's second: 2.06 ("Be fair and truthful…) should instead be "Be fair and avoid deception" because, as it stands, it seems only to prohibit falsehood. The revision would make it clear (as, apparently, "fair" does not) that a software engineer should also avoid deception by "not disclosing information, in other words, deception by omission."[6]

The next set of comments arrived one day later, October 12. It was the first to mention *Computer* as the source of the code. The respondent, a member of both the ACM

and the IEEE-CS, wrote from Cedar Rapids, Iowa, that he "generally" favored Version 3.[7] But he did have "some trouble with the wording of 3.07 ["Refuse to participate in any decision of a governmental or professional body…]"—enough trouble to vote "uncertain" on that provision (and only on that one). He then filled the rest of the one-page email with an explanation of what troubled him. It was what had troubled Notkin about the same provision—except that the respondent in Cedar Rapids thought the provision too narrow as well as too broad. Limiting 3.07 to "governmental and professional" bodies would leave "a loophole for company based activities." He therefore recommended shortening 3.07 to something like "Avoid conflicting financial interests." He then gave an example of the situations that he meant this revision of 3.07 to cover:

> While I was contracting to a computer manufacturer, that manufacturer was asked to co-develop a product for a health care provider. Our allegiance fell among several employers, customers, and users. However, it was possible to render opinions on equipment and standards to each company involved without feeling bias for the companies who were not important at the time. Our group was placed into a trusted position and all participants knew the relationship. This sort of conflict may easily result when a single employer and product relationship is not established. This is often the norm these days.

The respondent from Cedar Rapids did not explain what troubled him about 3.07 or why it was not resolved by 3.06 ("Disclose to all concerned parties those conflicts of interest that cannot be reasonably avoided or escaped and aspire to resolve them"). Perhaps he did not think there was any reason to "aspire to resolve" them in the example he described. Why then did he want to broaden 3.07 (since broadening it as he proposed would make 3.06 pointless and rule out the very arrangements he described so approvingly)? Clearly, there was something wrong with 3.06 and 3.07, either individually or in the way they interacted, but what was not clear was what was wrong—and how to fix it. The problem was not logical, but something in the way (some) users would read the clauses.

Then, for almost two weeks, there was a disturbing silence—not only no comments but also no ballots. The instructions in *Computer* had been clear. All ballots and comments were to go to Gotterbarn. What explained the silence? One possibility, not a very likely one, was that no one else had noticed the code because all reference to the code, ballot, and accompanying explanation had, unaccountably, been omitted from the table of contents. *Computer* had already undertaken to correct the omission by printing everything again in the November issue—this time being sure there was a reference in the table of contents. And, of course, the *Communications of the ACM* would also be publishing the code, ballot, and explanation in November. SEEPP was now definitely a month behind schedule and, without a reasonable number of ballots, definitely in trouble. The Steering Committee would have little interest in continuing with the code without a strong vote from the membership. Three years of work would be lost. Yet, there was little Gotterbarn could do now but push on, hoping that the comments would soon pour in.

## 10.2 From silence to SEERI

On October 23, the first ballot showed up in a three-page email from Omaha, Nebraska (with contact information at a corporate address). A page and a half of the email consisted of a long double column: on one side, the clause numbers (without the principles); and on the other, SF, F, or one of the other answers the survey had allowed. Most of the clauses had an F after it (Favorable). Almost two dozen had SF (Strongly Favorable). There were six U's (Uncertain).[8] There were, however, two O's (Opposed) and even one SO (Strongly Opposed): 1.12.[9] Following the ballot were comments explaining one U (2.01) and all the negative votes (except for "8.08 Consider code violations inconsistent…"). Some explanations raised important points. For example, our Omaha respondent offered three reasons for voting against 1.12 ("Delete outdated and flawed data…")—the one Strongly Opposed:

> First, with respect to the data, flawed or outdated data can still be meaningful in other contexts. Second, although it is tempered with the "where appropriate" clause, the action of the statement has a direct impact on the data and places the Software Engineer in a precarious position. Data are the persistent artifacts of application. Generally, the user of the software or other individual should be the steward of data content, not the Software Engineer. Finally, if a statement regarding data quality is to appear in the Code of Ethics, it should place the Software Engineer in an advisory role to the steward of the data, not in a role to act upon the data. A more acceptable statement would be something like: "Promote understanding and awareness of the data quality and data management issues that surround the data that affect the product."

Clause 1.12 began with Version 2. The corresponding provision in Version 1 ("Assure that raw information used in software is accurate, derives from a legitimate source, and is used only in ways properly authorized") was not subject to any of these three criticisms. Our Omaha respondent (without knowing it) also objected to other ways in which Version 3 differed from Version 1 (without knowing anything about Version 1). For example: He proposed removing from 2.01 ("Disclose to appropriate persons or authorities any actual or potential dangers to the user, a third party, or the environment…") exactly what SEEPP's executive committee had added:

> Maybe it is just the phrasing of the statement, but I would strike the final phrase ("or merely know about") from this statement. I also would strike 'actual or potential'. With these corrections, I would have responded "Favor" to this item on the survey.[10]

The Omaha respondent seemed to have no objection to the substance of the provision, merely to its tone or perhaps just to its legalistic precision.

Not all of his comments were reserved for ways in which Version 3 differed from Version 1. Three of his comments concern ways in which Version 3 resembled (or was identical to) Version 1. The most interesting, perhaps, concerned 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").[11] This is (along with Principle 2) the Software Engineering Code's (somewhat weakened) version of what engineers call "the paramountcy clause" ("Hold paramount the public health, safety, and welfare"). In some form or other, it has become a standard feature of engineering codes both in the United States and elsewhere. The Omaha respondent nonetheless objected:

> I think this statement is problematic. I think that it is important that self-interest not subvert the interests of the employer, client or public; however, the statement is too ambitious when it promotes the public interest ahead of that of the employer. If the Software Engineer has adequately "prequalified" the employer or client and determined that the employer or client is not actively working against the public interest, then I suggest that it is the responsibility of the Software Engineer to advise the employer or client regarding any concerns relative to the interest of the public at large. In this manner, the Software Engineer can play the role of educating those with the ability to affect the public interest.

Since our Omaha respondent thinks of software engineers as mere advisors, it is hard to see why he opposed 2.07. I do not think he is suggesting that it is enough for a software engineer to point out the public interest to a client or employer while recommending a course of action contrary to that interest.[12]

While most of the suggestions of our respondent from Omaha seem intended to weaken or at least soften the tone of Version 3, one does not.[13] His objection to 4.07 ("Inform when the project becomes problematic…") is that it is "too weak". He thought that, "[in] addition to informing, the Software Engineer should provide additional substantiation for his or her opinion." It is important that the software engineer "maintain a documentation regarding the informing of the client or employer".

That was the last comment for a week. Then, on November 2, there was a brief email from someone concerned with "assistive technologies". After saying that "[this] looks like a great effort and I wish you well", he quickly got down to business. He wanted to amend 2.05 ("Endeavor to produce software that respects diversity…") to include "sensory access" or "sensory abilities" (among the issues specifically mentioned "language, different abilities, physical access, …"). As 2.05 stood, he reported, he had "to read very broadly to understand that 2.05 includes the needs of the visually impaired and the hearing impaired." He concluded with thanks and the hope that "when I review this more in detail, perhaps I'll send you more comments.[14]

Almost another week passed before the next comment arrived, this one a page long from a young faculty member in Computer Science at the University of North Carolina. He was careful to say that the comments were on Version 3.0 "as published in the November 1997 issue of *Computer*". He then made ten comments, each signaled by the reference number in the code. Two were simply suggestions for amendment (with specific wording but no explanation): for both 1.05 ("ensure appropriate methodology") and 4.02 ("Ensure…document…approved for use"), "add: except with the knowledge and consent of all parties." Some suggestions indicate a problem as well as offering an amendment, for example "1.07: Again [as in 1.06], this is management's job. If the word "Ensure" were changed to "Provide", then I'm happy. Otherwise, I strongly disagree." [15] Some comments express doubts of one sort or another. For example, he said of 5.03

("Assign work only after taking into account appropriate contributions…"), "I don't understand what this means." In only one place did he seem to misread the code. Of 5.07 ("*Offer* only fair and just remuneration"), he wondered, "Why would I not accept whatever the hiring company wants to pay me, so long as the payment is not illegal and does not require me to do something unethical. What would be unfair renumeration [sic]?" Because he misread 5.07 (supposing it to apply to acceptance of remuneration rather than to offers), he said of the always troublesome 6:07 "Redundant with 5.07, and confusing for the same reason." He then took much of the sting out of these comments by ending his message:

> Good work putting this all together! This is a tremendous task, especially with all the opinions there will be from various people and groups. This has the potential to really state a direction and push SEs into more right-thinking action; perhaps even to modify behavior within software design houses to the benefit of us all. Thank you for putting in the hard work to help bring this about.[16]

Then almost another week passed without a comment on the Code. Gotterbarn had less time to worry about the silence than we might at first suppose. He was, of course, again teaching full time, but he was also setting up the "Software Engineering Ethics and Research Institute" (SEERI). SEERI was in part a way to promote the Code once it was adopted. Its website would carry the Code whether ACM's website or IEEE-CS's did or did not. He had already seen the advantage this would have. Most people seemed to have found Version 3 easier to access through Rogerson's (small) website at DeMontfort than through the much larger (and, therefore, harder to use) ACM or IEEE-CS sites. Gotterbarn also planned to use the SEERI website to list companies that had adopted the code (and thereby encourage others to do the same). The site would be a way to make available research on software engineering ethics—and on related topics such as licensing, course materials for teaching software engineering ethics, and ethics conferences. The website might even provide a place for ethics-related humor (such as Sam Redwine's quip, "Software and cathedrals are much the same. First we build them, then we pray!"). SEERI could offer seminars on software ethics, consult with individuals, organizations, and government, and even undertake major research on software engineering ethics.[17] East Tennessee State was so pleased with SEERI that it had freed Gotterbarn from the usual administrative duties of faculty and provided a graduate student to maintain the website and otherwise help with SEERI's day-to-day operation.

### 10.3 A positive response

While waiting for the flood of ballots to begin, Gotterbarn also wrote his task force (using the listserv) to announce that "work on the code must start again". This November 11 email reported the publication of Version 3.1 (and what had gone wrong in October), noted that the "articles" in both *Computer* and *CACM* contained "a survey for each item in the code" and that "responses are already coming in", and observed that "we should feel good about our product." But, he continued, the comments also "indicate that we are not done yet". There are "helpful comments" ranging from "things which are easy

to adjust to those which are not easy". Gotterbarn promised to organize the comments and "send them to you in a few weeks for your input."

Mechler responded (through the listserv) two days later (November 13, 1997). He had bad news: "In Computer Nov there isn't any reference in the Index. There is a Call for Participation on page 57." Even worse, "[the] web addresses on page 90 (survey) are not correct (same as on 57)." When he went to "What's New on ACM[,] Third one [presumably, Version 3] will not come up." Mechler ended by asking whether the survey is on a web page. Less than two hours later, Mechler answered his own question: "The third one is [at] the Centre for Computing and Social Responsibility [Rogerson's organization] and their correct site is http://www.ccsr.cms.dmu.ac.uk/[.] The article had it mistakenly as ccsr.cAs…" (The capital A indicates the error: it should be a small M.) Mechler concluded by reporting that the ballot "is not on any web page". Only the code is.

Gotterbarn immediately forwarded this bad news to the Task Force, preceding it with an unrelated request for help.[18] The Steering Committee had asked him for some responses from "corporate players, in addition to the individual responses." He was therefore asking any member of the task force who knew someone who could "speak for a company or a major area of the company such as systems development" to get the code and ballot to the appropriate person. Why the Steering Committee should have asked Gotterbarn to recruit corporate responses is not clear. The Steering Committee had several members who would at least seem to have had much better corporate contacts than SEEPP's executive committee (Cabrera, at Microsoft; Frailey, at Texas Instruments; Tripp, at Boeing; and so on). The same Steering Committee that only a few weeks before had sought comment directly from the TCSE and SIGSOFT now wanted Gotterbarn, at East Tennessee State, Miller, at University of Illinois-Springfield, and Rogerson, at DeMontfort University, to seek comment from corporate America. Was the Steering Committee also seeking such comments? Had they tried during the intervening month to get such comments and failed? Or were they just raising one more hurdle over which SEEPP would have to jump to bring work on the code to completion? Gotterbarn could only wonder and try to do what was asked.

Gotterbarn received the Steering Committee's request several days before he passed it on to the task force on November 13.[19] On November 18, Mechler emailed the appropriate person in his company, asking him both to respond as Gotterbarn requested and to pass on Gotterbarn's request to other chief information officers (CIOs) he knew. On November 20, Mechler's CIO wrote back with one name (and an email address).[20] Mechler then wrote (using the listserv): "One on the way CIO of Equitable Resources, Inc. Talking to another two." That was the last help Gotterbarn got from the task force in recruiting corporate response. By the end of December, Gotterbarn had several other corporate responses; some recruited from the contacts he was developing for SEERI, others just arrived unsolicited.

While Gotterbarn was making up his mind what to do with the Steering Committee request, he received a second ballot, this one with a Texas Instruments (TI) address (November 11).[21] The ballot had only four U's (and no O's or S0's): 1.12, 2.02, 2.05, and 3.08. There were only three brief comments. Though favoring 1.09 ("Ensure adequate testing, debugging, and review of software and related documents on which they work), our respondent at TI thought that "[ensuring] adequate testing and debugging

is partly the responsibility of the product provider but more so, the responsibility of those purchasing the product."[22] There is no explanation for the U vote on 1.12 or 2.02,[23] but concerning 2.05 ("Endeavor to produce software that respects diversity…"), our respondent at TI objected, "Targeting the audience of software users should be more of a concern/goal than trying to attack world-hunger with each software produced…it's a bit to[o] forward looking." What had brought on that reference to "world-hunger"? Was it the Code's explanatory sentence following the clause itself ("Issues of language, different abilities, physical access, mental access, economic advantage, and *allocation of resources* should all be considered")? Clause 2.05 was, Gotterbarn thought, one of the ways Version 3 had improved on Version 1. Another improvement was 3.08 ("Temper all technical judgment by the need to support and maintain human values"). That clause, the only other about which the respondent at TI had anything to say, elicited his most negative comment: "This principle seems too ambiguous. I cannot envision restricting technological judgment on the basis of any ethical situation."

The next set of comments to arrive was the first to mention the November issue of *CACM* as the source of the code. Its author, a principal in his own consulting service in San Antonio, began by "applaud[ing] the inclusion of language that stresses the importance of properly testing software systems prior to release." Indeed, he generally liked the code. But there was one provision he did not like, one of the innovations of Version 2 that had already come in for considerable criticism, clause 2.05 ("Produce software that respects diversity…"). The respondent from San Antonio devoted a page and a half to his five reasons for deleting the clause: "1. It's vague…. 2. It's contradictory…. 3. It's pointless…. 4. It's dangerous…. 5. It's unnecessary…." He then concluded with a paragraph that epitomized the whole:

> In Stalinist Russia, few examples so capture the fallacy and stupidity of socialism as "Socialist Science". Scientists bent their knees to the Communist Party and its tyrants, pursuing all sorts of ridiculous theories because they met with the approval of the Soviet hierarchy, turning their backs on the truth. I don't want to suggest that this subsection is on the scale of that grotesquery, but I do want to point out a parallel: All science must be ethically and morally informed, but it must not take sides in political matters. Of course, I grant anyone the right to view respecting diversity, however one defines it, as a pillar of one's personal ethics; however, I ask that this code not make it a mandatory part of the ethical values of every software engineer.

Clearly, clause 2.05 was in trouble.

The next day two more ballots arrived with comments, as well as some without; the day after that, a very long response, as well as some ballots without comments; the day after that, two more ballots with comments, as well as some without; and, before long, the tickle of response had become a small stream (about 40 ballots in all).[24] By this time (November 18), Gotterbarn (with the help of an honors student his department had assigned him for the purpose) had organized the comments into a single document, thirty-one pages single-spaced.[25] Gotterbarn sent it to the task force (in two parts) on November 20 (using the listserv), with a cover memo explaining what to do with it. After the salutation ("Hi Team"), the memo acknowledged "Ed's email about Industry input" and

reported that Gotterbarn had "contacted some multi-nationals, Lockheed Martin, TRW etc and we are getting statements from them." Gotterbarn again asked the task force to find corporate leaders (a "CEO") to respond, stressing that any such leader should "indicate that they are a CEO". He stressed this, he wrote, because "[he thinks] the steering committee's intent is to specifically identify the industry view." If the CEOs do not identify themselves as CEOs, how is the Steering Committee to know it got the industry view? He then turned to the comments (with the introduction, "Time to get back to work :-)" in capitals).

Gotterbarn had for some time been getting "responses from materials in the two magazines and on the web". Some are "useful and others are not so useful". He had included everything. The presence of a comment in the document "attached" (meaning, in those last days before attachments, below) did not "indicate support of the comment". Overall, the vote on the code's provisions had been "very positive" (approval by "90-100%"). But a few clauses (six) had not done that well (that is, had been opposed by "20-25%" of those voting). He then listed the six and commented on each:

> 1.12, - I think the issue is the SE cannot just delete data
> 2.05, - this is just a hunch—in the ballot the phrase "respects diversity" is, in the US, a politically loaded phrase[.]—This is evident by one two page critique of this clause. [The commentator from Texas had made his point.]
> 6.03 ["Support followers of this code"], Many uncertain votes (rather than oppose) put this in the 75% range—no[t] sure what the problem is.
> 6.07 ["Only accept appropriate remuneration"], There is a problem with this, we might just drop it.
> 7.02 ["Review others work only with their consent"], No idea what is wrong.
> 7.08 ["Not undermine another software engineer's job prospects for one's own personal gain"], We need to make clear that this does not rule out appropriate competition. We thought the word "undermine" would carry that weight.

What is remarkable about the provisions in trouble (apart from their consisting entirely of clauses added since Version 1 or much revised since then) is how few there are. Support for most of the code was overwhelming.

The problem, if there was a problem, seemed to be the form of the code. The chief problem was size. The most silly comment on size was, "Too long, too wordy. Can't be written on the back of an envelope." (Why would anyone want to write a code of ethics on the back of an envelope?) But there were more sensible versions of this objection, for example, one suggesting "a goal of reducing the code size to 40% of the draft size!" Some of those suggesting a shorter code seemed to do so because they thought much of the content was not about "ethics"—or, as one commentator put it: "Just because someone thinks it's a good idea does not make it an appropriate item for a code of ethics." Gotterbarn had a suggestion for dealing with this sort of comment without giving up anything: follow "the ACM model" by taking "the Eight keyword Principles and expand them by a word or three so that they could stand on their own" and attach "the clauses under each principle as 'explanatory detail', 'guidelines', 'specifications :-)'—we need a good word here." It would then be possible to print the code "in a short version, when necessary" and still "carry the sense of the whole document."[26] Having done that,

"why not call the document: Code of Software Engineering Ethics and Professional Practices[?]" After urging the task force to read the comments, especially those that "may seem ludicrous", and to suggest language to deal with them, he quoted his favorite comment on Version 3's size: "My initial reaction was 'major overkill', but I was wrong. I like it! I might even call myself a software engineer!" Gotterbarn sent the email through the listserv late in the afternoon of November 20.

10.4 How long, O Lord, how long?

The cover memo to this November 20 email had a "PS" (in capitals): "Please do not use the list address for personal correspondence!!!!" The PS seems to have been a response to an email exchange that had begun, early that morning, when Mechler had sent the listserv a one-word message "Test". An hour later, Manny Norman had responded (through the listserv): "In case you want to know—test received!" Five minutes later, he again used the listserv to announce: "By the way, I am receiving two copies of everything you send me. You may have me down twice in the database [because he had two addresses]." Norman then instructed the listserv to use his preferred address. Almost two hours later, Mechler responded (using the listserv): "First it was a test of the site; I think it was done this morning. Second, you must be on the site twice if you get more than one; I sent it to the site only. How have you been? Haven't heard from you for a while." Norman responded an hour later that he was being kept busy "juggling between performing OpenVMS system administration and programming, learning how to do same with NT, and teaching C++," adding that sometimes "I even go home! No rest for the wicked. How's yourself." Apparently, what began as a test of the listserv had unintentionally turned into a personal conversation broadcast to the entire listserv (because a "Reply" to the listserv functioned as a "Reply All").

The Mechler-Norman conversation seems to have had effects beyond Gotterbarn's postscript. Early next morning (November 21), Mark Kanko wrote the listserv—with the subject heading "Re: test": "Someone please remove me from this mail group." Even earlier that day, Tom Jewett had written Mechler and Norman (with a copy to Gotterbarn) with the same subject heading as Kanko:

> This is a large listserv, and as much as I'd like to meet everyone on it and be able to visit electronically, my inbox overfloweth even without multiple copies or test messages and personal comments that aren't meant for me. (Please note that this is NOT going to the listserv.)[27]

About 9:05 AM, Mechler responded directly to Jewett (with copies to Norman and Gotterbarn), "Sorry we inadvertently hit the reply and slowed down the large amount of traffic on the list." He then explained that he "was sending a test over and over because my e-mail kept coming back [observing that, thanks to Jewett's response, he knew that at least one person got the test]."[28]

An hour later Mechler wrote the listserv again, this time on official business, though with the salutation "Don". Mechler had not yet read "all of the attached comments" but already wondered "with such a high rating for the present form why change it now?" Changes meant "all the reviews again". Would it not make more sense

just to work on those provisions that received "negative responses"? Mechler was here exhibiting "engineering conservatism" (sometimes crudely summarized, "Don't fix what ain't broke"). But he was also identifying a deeper problem with the SEEPP process as it had developed since Melford's departure almost a year ago.

Until Melford left, SEEPP had tried (however unsuccessfully) to follow its "Guide to Operations".[29] The Guide had a well-defined process for developing "standards" (by which the Guide meant rules written, like the code, with "shall"). The working group's chair was (according to the Guide 4.1.3 and 4.1.4), to use "as expeditious a method as needed to establish the WG membership's consensus" on whether "the completed draft can subsequently serve as the foundation for an iterative process of refinement." The working group did not have to agree with every part of the document, or even be willing to have it adopted in its present form. All they had to agree on was that the document was good enough to work with. Mechler had done that with *his* "sub group" for Version 1. While Gotterbarn had implicitly agreed with Mechler's group that Version 1 could serve as a working draft, with Version 2.1 Gotterbarn achieved consensus again within his own working group (by then, also the task force).

Once there was consensus that a document could serve as the basis for drafting an acceptable standard, the next step was to enlarge the working group into a "balloting group". A balloting group had to have "balance" (that is, "[avoid] dominance by any single interest category "). The balloting group obtained balance by adding to the working group "all other appropriate individuals and organizational representatives." (Guide 4.3) The ballot put before this group was to have three choices: Approve, Do not Approve, and Abstain. Any negative vote was to "be accompanied by specific reasons in sufficient detail so that the specific wording of the changes that will cause the negative voter to change his or her vote to 'approve' can readily be determined." If the changes are made, then the vote "automatically becomes affirmative." In the absence of reasons for a negative vote, the ballot "shall, after a follow-up inquiry, be classified as 'no response.'" An Abstain counts (more or less) as a "no response".[30] At least three-quarters of the ballots must be returned (abstentions counting as returns), but more than thirty percent abstentions invalidates the balloting. The working group was then to resolve as many of the negative votes as possible (by accepting the changes proposed). Those that could not be resolved, "together with the reasons of the negative voter and the rebuttal by the WG members conducting the resolution of the ballots, shall be submitted to the balloting group, providing each member an opportunity to change his or her ballot." The Guide even recognizes that "[further] resolution efforts may be required if additional negative votes result." The Guide's process puts the pressure on the individuals objecting to find a way to agree. Where "a significant number" of individuals continue to object to some provision, the task force may recommend the standard for "trial-use", that is, for temporary adoption to see whether it can prove itself in practice. (Guide 4.3.6) Every step had a deadline. There was no endless recycling.

Gotterbarn did not answer Mechler's question ("why change now?"). Instead, an hour after Mechler posed it, Gotterbarn responded to "Ed and every body else" (using the listserv) that the question indicated that "my request to you all may not have been clear". So, Gotterbarn is not "broadcasting my reply" (a frown sign following, indicating unhappiness, whether with having to write again, or with the exchanges earlier in the day about using the listserv for messages directed to individuals, or perhaps both). Admitting

(once again) that yesterday's email (the collection of responses) was "VERY large", he tried "briefly" to clarify his request (the one in the cover memo). He wanted responses from "industry folk" identified as such. He wanted everyone to read the comments (noting that not every clause received comment). If a comment seems "on the mark" (and only then), "indicate how you agree with the comment" and also "suggest some re-phrasing". He also wanted everyone to look over the Principles to see "if we can make SLIGHT modifications to them" to pick up anything important in their subsidiary clauses (so that they can "stand alone" and still "convey the sense of the code"). If "we could do this then we could both keep the Code as a whole—Principles and Clauses[—] and print the Principles on the back of a society's membership card! :-)." After again urging members of the listserv "not to Broadcast letters that are intended for individuals", he concluded, "We are in home stretch and the results are very positive!!"

While Gotterbarn was waiting for his task force to respond, comments and ballots continued to come in. Perhaps the most unusual of these late arrivals was from the Planning and Architecture Office of St. Paul Companies (of St. Paul, Minnesota). Beginning with a very business-like "Sir", the email indicated that St. Paul "would like to adopt the ACM/IEEE code for our internal use." St. Paul had "modified it slightly to generalize the language beyond systems engineers." The writer wanted to know what sort of "source reference or citation would be appropriate". St. Paul had already been working on an internal code when it saw Version 3. Its code would gain "considerable additional strength when backed by these two prestigious societies", but it had found Version 3's wording "cumbersome for employees who are not [in] a consulting arrangement, and for IT staff that are not doing systems engineering". He would send Gotterbarn the "'modified' draft" with the revisions clearly marked.[31]

Thus began Gotterbarn's first experiment with introducing the code into a business environment. He responded "yes" (or "of course") to St. Paul's request for permission. When he contacted St. Paul's months later, he learned that the company "had only incorporated sections of the code into [its final document]." This was a success compared to some companies he dealt with later. There seemed to be two major blocks to a company just adopting the code as its own. One is (what Gotterbarn came to call) "the dinosaurs problem". A company officer who originally wanted to have the company adopt the code for its software engineers would eventually report back that the company could not "officially adopt the code" because "getting it through their bureaucracy" would be "too much of a hassle". (The "bureaucracy" made the company a "dinosaur" unable to adapt to a new environment.) The other problem was corporate counsel. Often the company's lawyer would block adoption because the company should not be "supporting one professional society rather than another." The corporation should be neutral between professional societies; and adopting one code rather than another (even if the codes covered different activities) might constitute support. One corporate lawyer raised an even odder objection. The company could not *publicly* adopt the code. The company in question manufactured computers and had a policy against revealing the source of parts used in constructing its computers. The lawyer thought that the same policy should apply to the source of the company ethics policy. The company could adopt the code, but they could not publicly admit to having adopted it.[32]

For ten days, no one on the task force responded to Gotterbarn's November 21 email—and no one else responded to the November 20 email either. Then, on December

2, Gotterbarn received two responses, one from Fairweather in England, the other from Weckert who, "still in Spain" (rather than at home in Australia), was (he apologized) in no position to get responses from industry. Both emails were long (six to seven pages). Weckert's response was all detail. He seemed to have gone through Gotterbarn's collection of comments, cutting out those he wanted to respond to, pasting them in his email, and then inserting his response in caps. A typical response would be, "Perhaps, but I'm not sure where it should be" (in response to a suggestion that the terms 'public', 'employer', 'client', and 'user' be clarified).

Fairweather's introductory sentence nicely sets the stage for the comments that follow: "My overall feeling is that if we are getting the high levels of approval that we are, we should 1) be wary of making changes that some who currently approve of the code might disapprove of; 2) be wary of major new developments such as stating the principles in a 'stand alone' manner." There followed pages of detailed response to particular changes Gotterbarn (or, more often, one of the respondents) had suggested. Of special concern to Fairweather was the short version of the code Gotterbarn had suggested. While wary of it, he thought it "would be OK **provided** we make it clear that they [the Principles standing alone] are only extracted from the full code, and should never be used on their own, or quoted without a reminder that they are extracted from the full code." Fairweather even took the trouble to suggest the form in which he thought the extracted Principles should be printed. Of special importance to him was that the in-particular clause not be omitted. The presence of an in-particular clause for each Principle would make clear that the Principle could not really stand by itself.

## 10.5 Three days in Baltimore

Gotterbarn was now making final preparations for a meeting with the other two members of his executive committee to revise Version 3 (or, as he now admitted, "build Version 4"). The meeting would be in Baltimore. An important step in those preparations was incorporating into the comment document all comments that had arrived since November 20. That work was completed by December 1 (the day before the Fairweather and Weckert emails arrived).[33] The final summary, though a few pages longer, contained nothing strikingly new. The comments had become predictable.

By December 3, Gotterbarn had two other documents ready. One was the final tally of ballots. Using Excel, Gotterbarn was able not only to tally the votes, but to calculate percentage in favor and opposed to each clause. Overall, the mean approval had been just over 94% and the mean disapproval just over 3% (with the difference in uncertain votes).[34] Four clauses (1.10, 1.11, 3.02, and 3.03) received unanimous support; and one more (3.06) passed without any votes opposed. Software engineers seemed to have no doubt that they should promote the privacy of individuals (1.10), use data legitimately (1.11), only sign documents within their responsibility (3.02), and reject bribery (3.03), and were only slightly less certain that they should disclose conflicts of interest (3.06).

Having completed the tally of ballots, Gotterbarn could prepare what he jokingly called, "Version.3b mod-notkin, et al").[35] This was the Version 3 that the executive committee would work from. The first two lines, inserted above the title (in large print, bold, italics, *and* underlined), explained the format: "This has some of the modifications

[to consider] and clauses in bold have less than a 90 [%] favor rating." Twenty clauses were in bold (a quarter of the Code's eighty clauses).[36] Below this heading (in bold) was the new title of the code ("Software Engineering Code of Ethics and Professional Practices") and, immediately below that, the same title again. This was the first of the "modifications" already incorporated. Next in bold was a question, "Does this footnote get attached to the CODE?" The footnote, listing the codes "referred to", had, until then, been part of the article introducing the code. Why Gotterbarn thought it should be part of the code is unclear. No one who commented on Version 3 had suggested it should be and, of course, footnotes of that sort are rare in codes of ethics. (Indeed, I know of no code that has a footnote listing source codes.) Here perhaps the scholar had gotten the better of the draftsman. (The footnote did not survive Version 3b.)

All the modifications seem to be responses to Notkin.[37] The first (in caps) is just under the title "Preamble": "Told Notkin that this was going in the Preamble—where." "This" consists of two sentences:

> Software engineering is unique in the amount of time spent in maintaining and modifying existing software. The flexibility and maliability [sic] of software systems places [sic] special burdens on the software engineer to treat software modifications with the same professionalism as new development.

The first paragraph has two places where "and maintenance" (in italics) has been inserted after "development", with other changes that Gotterbarn had promised Notkin scattered here and there. The largest of these was in 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, *provided this does not violate confidentiality, or cause loss of trade secrets, or cause competitive disadvantage (see 4.05).*" (This clause—without the addition indicated by italics—had received an 89% favorable vote, just under the 90% cut off.) There is only one surprise. Just above clause 3.07 ("Refuse to participate…") Gotterbarn wrote (mostly in caps), "Early returns had some objections to this/now sits at 95% positive! Told Notkin it was gone. Ooops."

There is a lesson here. Notkin had (rightly) said that he spoke for himself. He had made a prediction about what his technical council would do, but seems to have badly misjudged it. In fact, the council seems not to have cared enough one way or another to respond. Notkin had never had to test his own ideas in open debate. He had supposed that he spoke "common sense" (and Gotterbarn had thought so too). But the December tally at least suggested that common sense may not have been what Notkin supposed. Leaders generally suppose they know what "the profession thinks". This was, it seems, what Shaw and Frailey thought too. They might have been right—but, in fact, they were, it seems, largely wrong. There are nonetheless at least two reasons to take the views of leaders into account. One is that they may have a better understanding of the issues than their followers; another is that they may have the power to block a code even if they are wrong in their views, dogmatic, and dense. The first reason is always worth considering; the second, something to get around if possible, for example, by taking a vote that reveals how wrong the leader is in a way not easily dismissed. Gotterbarn would have done well

to promise Notkin less, especially since he did not (it seems) send his official response to Notkin until most of the ballots were in and tallied.

The next day (December 4), Gotterbarn, Miller, and Rogerson met briefly in the lobby of the Brookshire Hotel in Baltimore. It was late afternoon. All of them were tired after traveling most of the day. Each had a laptop computer with him. Gotterbarn distributed copies of the documents he had prepared (the tally of votes, the final collection of comments, Version 3b, and the emails from Weckert and Fairweather). Some of these were on a diskette; others on paper. After some discussion of large issues, division of labor, and timetable for the next three days, the three agreed to meet again in Gotterbarn's room early the next morning, adjourned to dinner, and retired early.

The meeting's location was not arbitrary. Gotterbarn had chosen Baltimore because traveling there was relatively easy for all three, because its hotels were relatively inexpensive, and because each would be far from the distractions of home. Gotterbarn had chosen the Brookshire because it had all the conveniences of the city close by, was not too expensive, and had two-room suites. The outer room of the standard Berkshire suite had a work table (or large desk)—as well as the usual couch and comfortable chairs. Only Gotterbarn's choice of meeting room was arbitrary. Miller and Rogerson each had a similar suite down the hall from Gotterbarn's.[38]

The three (as agreed) met in Gotterbarn's suite after breakfast. They were to go through Version 3b once, making such changes as the comments or tally suggested; then to go through the revised document, making sure that it said what it should and, when it did not, revising accordingly; and so on until they were satisfied. Nothing was immune from improvement. Once work on the code was complete, they would draft two letters: one to the task force explaining what they had done; the other, to the Steering Committee asking it to recommend Version 4 to the two societies for approval. Gotterbarn expected the work to take two full days. In fact, it took much of two evenings as well—and much of the third day's morning. Rogerson, who volunteered to be the recording secretary, sat at the table. Sometimes Gotterbarn and Miller sat at the table too, looking at a paper copy of this or that document or amending their own paper copy of Version 3 when they had agreed to a change. Among the pile of papers on the desk was one containing comments (and ballots) that had been faxed. These (unlike the emailed comments) were generally short (and handwritten).[39] Sometimes, though, Gotterbarn and Miller stood behind Rogerson, looking at what he was typing, or took turns pacing the oblong room, sitting on the couch or in one of the chairs. Though the room had a big window at one of its short ends, little day light came in. (Gotterbarn had pulled the drapes to make reading the computer screen easier.) Discussion was almost continuous. As Miller recalls,

> Several times we got into deep discussions about the wording of a clause. I think all three of us felt that what we were doing might have significance beyond what usually occurs in academic scholarship. We wanted to agree to something (to get it "out the door"), but we wanted to get it right. Some of the suggestions we had gotten about the code made it clear that every word counted --- there were shades of meaning in what might otherwise be considered trivial word changes.[40]

On the first day, Gotterbarn put off lunch until the middle of the afternoon saying over and over, "We're making progress. Let's just finish this first run through, then go to

lunch." Gotterbarn treated toilet breaks in the same way—until Miller revolted, declaring he was going "now" no matter what. Only when keeping track of changes to their paper copies of the code became too hard did Gotterbarn agree to lunch, and then only on condition that they stop at a copy shop to have the code's electronic version put on paper. Having eaten a quick lunch, they found a Kinko's. Kinko's copiers had only recently been put on a network that allowed printing from a diskette. The first attempt to print failed. The staff, well-groomed clerks in their early twenties, were not sure what was wrong. SEEPP's executive committee, with no time to waste, volunteered to fix the problem. The staff hesitated, perhaps worried what the three, all obviously survivals from the pre-electronic age, might do to their sophisticated system. Reason, firm looks, claims of expertise, and a few well-placed technical observations won out, however, and the three software engineers were soon checking wires and adjusting settings until they identified the problem, a cable that had come unplugged. The network fixed, they quickly printed three copies of the nascent Version 4.0 and returned to their work at the Brookshire. Miller thought it "was somehow fitting that we old geeks figured out what was wrong at Kinko's. We were going to get that code finished no matter what!"[41]

Dinner, though quite late, was pleasant. The three had a first draft of Version 4. The end was in sight. Tomorrow they would work on the details. Next day, they worked much as they had the day before—but when they went to dinner that evening, they had everything they had hoped for in the code. The next morning they completed work on the two letters. By noon of December 7, the meeting was over. They had checked out and were on the way to the airport. Reflecting on what they had done, Gotterbarn observed, "The most effective way to do it [this sort of major revision] was NOT by email from separate distracting environments. Face to face—mind to mind—working in the same direction gets the work done much better and more efficiently."[42] The three days in Baltimore had had much the same feel as those months at DeMontfort. Gotterbarn was (as he put it) "in his glory" during those three days, but for several years afterward Miller (with a smile) referred to Gotterbarn as "the taskmaster".[43] Such is the price of glory.

Gotterbarn flew out of Baltimore early in the afternoon of December 7, getting home quite late. The next morning (December 8) was the beginning of exam week. He had tests to give and projects to grade. Almost the first thing he did that morning was to send the task force Version 4.0, with the cover memo (signed "don") that the executive committee had prepared the day before. After declaring that "the results of several emails and a three day meeting by the executive committee" were "below", Gotterbarn noted, "We need to send the FINAL COPY OF THE CODE BY MONDAY DECEMBER 15." He was asking the task force to "give this one more review". The "substance of the code" was unchanged, but "we did clear up several things and restructure it." He then summarized what the executive committee had done. First, they had "modified the Principles so that they can stand alone as a reminder of the entire code". The short version now stood before the main code. Second, they had shortened the Preamble, but "also included issues of maintenance", and had tried to make it clearer "when resolving ethical tension that the public interest is paramount". Third, they had re-ordered the principles, putting "the PUBLIC first where it belongs". Fourth, they had reordered the clauses "in general putting the more aspirational elements first and the more specific items later". Last, they "clarified the function of the code by renaming it". Gotterbarn concluded by again pleading for "your comments".

Though this summary of changes makes it clear that Version 4.0 differed substantially from Version 3.0, it in fact substantially understates the differences in at least five ways. First, the executive committee had shortened the Preamble, in large part, by deleting Version 3's second and third paragraphs. All references to "three levels of ethical obligation" were gone. Comments had shown that the distinction confused rather than clarified. The idea of a code as consisting of "aspirations", "expectations", and "demands" had (at least officially) been abandoned.[44] Whatever its theoretical value, the distinction had no practical value within a code. As Gotterbarn explained years later:

> In version 3 I had tried to meet some of the objections we had heard early from the steering committee- what makes this a SE Code of ethics when it says some things said by other profession's codes? So the preamble contained comments on three levels of ethics—common human virtues, professional ethics having due care, and specific professional ethics related to a particular domain. It was this latter part that made it an SE code. Anyway, this distinction caused confusion because people reading the preamble expected to find each clause clearly labeled as to which of these three levels it was on. We knew that any attempt to deal with this type of [objection by] dividing the Code would have landed it in a mire of debate about which clause belonged where and would have distracted from the intent of the Code. One of the good things we did at that meeting was to scrap my three-level stuff.

What the executive committee meant by re-ordering the clauses so that the "more aspirational" ones come earlier is not clear. In what way, for example, is 1.01 ("Accept full responsibility for their own acts") more aspirational than 1.08 ("Volunteer professional skills to good causes…")? Is this another example of the concept of "aspiration" doing no work?

Second, the executive committee had not limited changes in the Code to clauses with a relatively high percentage of opponents. They had in fact made many small changes in clauses with very high approval ratings. For example, they had replaced "employee" with "software engineer" in both old 5.01 and 5.02 (new 5.02 and 5.03). Both clauses had been approved by more than 95% of those voting.

Third, the executive committee had also made larger revisions in clauses with equally great (or greater) support. The most dramatic of these revisions is the new 4.04 ("Do not engage in deceptive financial practices such as bribery, secret payments and double billing"). This clause replaced *three* clauses in Version 3: 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"); and 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). Clause 3.03 had a 100% approval; 3.04, 98%. Only 3.05 (at 89%) had failed to achieve the 90% that (in Version 3b) declared would mark a clause for revision.

Fourth, while this one revision may seem to explain the drop in number of clauses from 80 to 78, it in fact does not. Several other clauses had entirely disappeared (such as old 2.03 "Affix their signature only…" or old 7.08 "Not undermine another software

engineer's job prospects...).[45] Others (unrelated) had replaced them, for example, new 3.15 ("Treat software maintenance with the same professionalism as new development"), inspired by Notkin, or new 5.12 ("Not punish anyone for expressing ethical concerns about a project"), which, though reasonable, does not seem to have come out of the comments at all.

Fifth, Version 4 was about four-fifths as long as Version 3.[46] The change is actually greater than this fraction suggests. After all, the one-fifth reduction was achieved while adding the (largely redundant) Short Version (189 words). Without the Short Version, Version 4 would have been just under three-fourths the length of its immediate predecessor (though still almost twice the length of Version1). A reduction in length of one quarter, especially when combined with the addition of new clauses, is independent testimony to the scale of revision accomplished in the first week of December. Version 4 is almost a new code.

The executive committee had, it seemed, not felt at all bound by the poll of ACM and IEEE-CS members, by the drafting principle "Leave well enough alone", or by Mechler's concern that making any but minor changes in Version 3 would bring on another poll, putting off adoption for many more months (and thereby chancing a wreck on shoals already perilously close). Whatever improvements the executive committee had made (and there were many), they came (as the committee should have learned from the large dissent on "diversity") with the risk that what they thought uncontroversial improvements might prove quite controversial. They were, in effect, gambling the overwhelming support the Code had achieved.

Re-ordering the Principles and the Clauses under each Principle also had the effect of making tracking changes, and comparing changes and votes, much harder for anyone who tried (including the executive committee). That effect, though doubtless unintended, is worth noting, especially since there was not much reason to reorder the Principles (only one commentator suggested it) and no reason to reorder the Clauses (since no commentator had suggested that). Even someone who liked the re-ordering might complain about that effect. For example, Fairweather wrote that the "[re-ordering} was good, but in the process it becomes more difficult to spot which clauses have been deleted, and some of the ones deleted, I feel, ARE important." [47]

## 10.6 First comments on Version 4

Gotterbarn had asked for comments when he sent Version 4 to the task force on December 8. The first comments to arrive were mine (on December 9), a three-page fax (not counting the cover).[48] My "overall comment" set the tone for the particulars that followed: "very soon, it [the code] should be gone over by a good editor." To soften the implied criticism, I added in explanation: "Committees have a tendency to garble text, to introduce unnecessarily clumsy modes of expression, and otherwise to make the code more difficult to use than need be." The comments that followed this explanation were (I said) intended as illustrations of what I had in mind; my silence on substantive issues indicated only "my virtually complete satisfaction with the Code's substance". I said nothing about the reordering of the Principles or clauses because, though I disapproved, I regarded order as a matter of taste. There are many ways to order a code. If the likely

users could not see the advantages of the order I chose, they were welcome to anything they thought better.

My objections were of another sort. The revisions had introduced what in software are called "bugs", at least three kinds of them. First, I had found a great number of simple errors (typos)—indicating nothing more than haste. For example, consider my suggestions (given in brackets) for the last three lines of the Preamble's fourth paragraph: "In these judgments concerns [delete s] for the health, safety, and welfare of the public is the primary concern…. That is [insert ,] the 'Public Interest' is central to this Code."

Second, I had noticed sloppy drafting. The first instance (occurring both in the short version's introduction and in the corresponding sentences in the long version's Preamble), was a "transition from 'must' [in the first sentence] to 'shall' [later in the paragraph]". Whatever the point of the transition, it is (I said) "too subtle for me". I therefore recommended "using 'shall' in both (since 'must' sounds too much like banging on the table)." Changing terms without a clear change in meaning invites confusion. Does the change mean something? What? (The next edition of Version 4, and all Versions 5, followed my suggestion in the Short Version but not in the Preamble.)

I also found several places where, though there was no risk of confusion, there seemed to be unnecessary redundancy. For example, I suggested revising the last sentence of the Preamble's fourth paragraph ("That is[,] the 'Public Interest' is central to this Code"): "delete this entire last sentence as entirely redundant, enough [in the sentence before] to say the public health, safety, and welfare are primary." This suggestion was not taken.

Third, Version 4 seemed to have introduced a number of conceptual problems. Consider, for example, the new 4.04 ("Do not engage in deceptive financial practices such as bribery, secret payments and double billing"): "Is bribery a 'deceptive financial practice'? [I asked] What if the bribery is open?" It might be better to say, "not engage in bribery, double billing, or other improper financial practices." I had, I added, "[omitted] 'secret payments' since some—for example, those involving national security—might not be improper."[49]

Another one of these conceptual problems was in new Principle 1. It now read, "Software engineers shall always act in the public interest…" I suggested that it was enough if they always acted in ways "consistent with" the public interest. "To require them always to act 'in the public interest' is to forbid them to do good for themselves, their clients or employers, and even customers when doing that is neutral with respect to the public interest." Surely, that was not what was intended. Indeed, that suggestion would be inconsistent with Principle 2 ("Software engineers shall act in the best interests of the client or employer within the confines of the public interest.").[50]

Gotterbarn seems to have received only one other response from the task force before Version 4 was due at the Steering Committee. That one, by direct email, was from Ben Fairweather—like me, a philosopher, not a software engineer. What he wrote was, as he put it, merely "IMHO" (in my humble opinion); it was almost entirely substantive. Apparently, Fairweather had not only read through the document carefully, he had, despite the difficulty re-ordering introduced, systematically compared each provision of Version 4.0 with the corresponding item in Version 3.0. He had much to say about Version 4.0, some negative but much of it positive. Quickly, without comment, Gotterbarn forwarded Fairweather's email to the listserv.[51] Later that day, Rogerson

(Fairweather's boss) replied directly to Gotterbarn, with a copy to Miller. A few hours later, Miller also responded.[52] Miller seemed a bit grumpy. His "initial reaction to all these comments" was that "we cannot do this forever, trying to please everyone over and over." Having made that point, he relented: "if we can QUICKLY come to some language that is a clear improvement on our already improved language, so be it." Miller did not have much to add, but what there was was inserted below the relevant part of the Fairweather-Rogerson exchange he had pasted into his email. The Fairweather-Rogerson-Miller exchange offers more insight into how the executive committee now approached the comments they were receiving.

Fairweather's positive comments showed the care he had given Version 4. Many were of the form: "Improved" (after quoting the new Principle 1 in full); or (after quoting 2.01) "An improvement on the old 4.01"; or most often (after quoting the new 2.04) "A mild improvement on the old 4.02". But some of the positive comments were more complex (and mix in criticism); for example, Fairweather admitted that the new Principle 3 ("Software engineers shall ensure that their products and related modifications are of the highest professional standards possible….") has been improved by the "removal of detail" but adds, that "loss of mention of products being 'acceptable' to the employer, the client, the user, and the public creates a deficiency with the code: it should have a clause [requiring acceptability]." [53] Rogerson's response was, "If they are of the highest professional standards, then by inference they will be acceptable as this is part of the definition of professionalism." Then, perhaps realizing the risks of relying on inferences from a definition of "professionalism" not given in the code (and perhaps controversial), Rogerson relented, "An extra clause (or addition to an existing clause) is worth doing." Miller disagreed "violently":

> If you want yet another clause, fine. I might point out, however, that you can work to the highest standards and still have some idiot not "accept" your work. Part of being a professional is knowing when to adhere to standards when idiots demand something else.

A good point, but one that admits the possibility that "the highest [professional] standards" might be too high for public, customer, or user (unless "possible" means something like "practical" rather than "technically possible"). The more detailed language of Version 1, which survived more or less through Version 3, avoided this problem: "Software engineers shall, insofar as possible, assure that the software on which they work is useful to the public, the employer, the client, and the user, completed on time and at reasonable cost, and free of significant error." Anyone who did not find acceptable a product that was useful in this way, completed at reasonable cost, and free of significant error would, indeed, be an "idiot". The new Principle 3 was not changed.

The most negative of Fairweather's comments concerned the newest part of the code: "I cannot accept this short version as it stands. If it is not modified, I will have to ask for my name to be removed [from the list of SEEPP members at the end of the code]." Fairweather then explained why he was unhappy (in effect, summarizing what he had written on December 2): "the lack of warning not to treat this short version as a complete code of ethics is a fatal flaw." (10.4) The flaw was, however, easy to remedy, just "add a line that it is not the complete code, and is only an 'aide memoire' for the full

version, which itself is not intended to be used so that individual parts in isolation could justify errors of omission or commission." Rogerson's comment was that "Ben's concern is valid". He suggested adding a sentence that "points people to the full code and explains the importance of the clauses."[54] Miller agreed and even suggested a paragraph that (without amendment) became the first paragraph of the short version of the next edition of Version 4.[55]

Fairweather's negative comments did not always receive such favorable treatment. Consider, for example, the discussion of 8.08 ("Encourage colleagues to adhere to this Code"). Fairweather objected that it "doesn't appear to me to be suitable for the 'self' principle" (and suggested going back to old 8.08 with a small revision). Recalling that I too had suggested that 8.08 belonged elsewhere (under Principle 6 Profession), Rogerson recommended instead that 8.08 be revised to read, "Consider violations of this code inconsistent with being a professional software engineer." Miller's response was, "I don't care." The next version of 4 adopted Rogerson's suggestion.

Rogerson was not always this accommodating, however. For example, Fairweather objected to the rewriting of Principle 5. The principle, while improved, had (Fairweather noted) lost "1) acting fairly (in general) and 2) 'enable and encourage those who they lead to meet their own and collective obligations, including those under this code'." Judging that loss to be "severely detrimental", he recommended adding "a new 5.01 [for acting fairly] and 5.02 [for enabling]." "Not convinced—leave as it is", was all Rogerson had to say. Miller agreed: "Yes, leave as is." It remained as it was.

Occasionally, Miller was quite dismissive. Fairweather thought it "detrimental" to delete from old 2.01 (now new 1.04) the words "or merely know" after "reasonably believe". Fairweather would accept deleting "merely". Rogerson replied that he thought "believe" includes "know" in this context. (1.04 read: "Disclose to appropriate persons or authorities any actual or potential danger to user, the public, or the environment that they reasonably believe to be associated with software or documents.") Miller responded directly to Fairweather (though Fairweather was not among the email's addressees and probably never saw the response): "Put a cork in it, Ben. What we have here is good." Clause 1.04 was not amended.

Over the weekend, Gotterbarn revised Version 4 (while also grading exams). Monday afternoon, December 15, he emailed the listserv a one-page memo entitled: "Version 4.DONE". The memo maintained the tone set by that title. Gotterbarn thanked "you all" and described the writing as "an exhilarating experience"—adding that "[when] projects come to a successful completion[,] there is normally a celebration and everyone shakes hands and is proud of the work accomplished." He expressed the desire to meet all those he had so far only known through email. "The problem with E-mail is that many of you only have a virtual persona for me." Then, getting down to business, he reported, "We have received several comments from the task force on Version 4.0." Some were positive, some were not so positive; some substantive, some concerned with "issues of grammatical preference". "We" had tried to take them into account in the "final version". Some of the lengthy changes might be better in "worked examples of the Code". (Apparently, Gotterbarn was here referring to one of Fairweather's suggestions.) He would, he said, explain those "worked examples" in a later email. The code was not perfect. But, as in any project, "at some point we must call [this one] complete even though we can still think of other improvements." He had decided now was the time.

Hoping the task force approved of what had been done, he noted that his memo was short "because I have placed the final version on a web page: http://www-cs.etsu.edu/seeri/secode.htm."[56] (SEERI was now in operation.) Task force members should read the code "one more time" and be sure to "carefully check the credits at the end of the Code" because he wanted "the form of your name and its spelling to be correct and to your liking." This version was the one "being sent" to the Steering Committee for adoption by both the IEEE-CS and ACM.[57]

## 10.7 Party time

The next morning (December 16), just before 9:00 AM, Manny Norman wrote the listserv (replying to "VERSION 4.DONE"): "Thanks, and the season's greetings to all of you. Perhaps, we could figure out some sort of get-together at some mutually acceptable geographical location sometime in the New Year." Apparently, Rogerson liked the idea. He quickly responded: "How about the Christmas Islands?"[58] Duncan Langford soon joined the horseplay: "Brilliant idea. Now, have we enough left in Don's budget to fund fares & accommodation…:-) Season's greetings to all!" About 11:00 AM Gotterbarn joined the discussion: "Budget ????  did I miss something?!?!" (Almost everyone who had anything to do with SEEPP knew by now there was no budget.) Having made a joke, Gotterbarn almost became serious: "There is a splendid conference ETHICOMP '98 in Rotterdam in March. The ACM is having an Ethics Conference in Washington in May. We can have two parties."[59] About the same time next morning, the horseplay began again. Mechler wrote, "I never thought e-mail could catch a sigh of relief but the last few days emails was [sic] a perfect example." Then, perhaps thinking this comment was not light-hearted  enough, he continued, "Isn't it just like a leader to ask what budget when the project is over and everyone wants to celebrate." Mechler wondered how there could be an ethics meeting "in Washington????" and wished everyone "Happy Holidays". A half hour later, Norman ended what he had started the day before—by a lisping pun: "Maybe we should all meet across the Detroit River from me—in Ethics (Essex County, Ontario!)." Then, perhaps thinking he had gone too far with the pun, Norman apologized: "I know we're not supposed to be using this listserv for 'flippancy', but I think we have reason to celebrate this time."[60]  Everyone must have agreed—or, at least, no one wrote the listserv to condemn the exchange. It was, after all, doubly a season to be jolly.

Before the horseplay was over, Gotterbarn sent the Steering Committee—or, rather, its chair and vice chair, the official ("final") Version 4 with a cover memo (just over two pages long).[61] Signed "The Executive Committee of the SEEP Task Force", this memo is itself a significant document. Not only did it summarize the task force's achievements in a way likely to impress the Steering Committee as a reason to recommend the code to the two societies for approval, it also announced plans for disseminating the code that seem not to depend on whether the two societies approved. The memo made no mention of the Steering Committee's plan to disband at the end of the year (a plan about which the Steering Committee had said nothing since September and which the executive committee doubtless hoped had been forgotten).

The cover memo's first paragraph offered five (unnumbered) reasons for recommending the code for approval:

- Support by both societies will be a public assertion by the profession of a unified commitment to ethical responsibility,
- The Code represents an international standard of practice,
- The Code has been reviewed and adopted by industry,
- The Code presents a broad consensus of the profession, and
- There is wide demonstrated positive support, including from Academia, for the Code.

The next few paragraphs provide detail in support of these claims. For example, to explain why the support of the two societies is important, the memo distinguished the software engineering code from those of the two societies. The software engineering code "addresses the ethical issues and standards of conduct which specifically arise within the practice of software engineering." It is not ("[in] this sense") a substitute for any existing code.

While stressing the international participation in the task force ("Europe, Africa, Australia, and North America"), the memo also pointed out other ways in which SEEPP had avoided too narrow a perspective. The task force had sought "responses from members of the major computing societies". It had "received [comments and support] from the International Federation of Information Processing (IFIP)."[62] It had even included "two lawyers [Barber and Phillips] specializing in computer law, and one representative from the military [Kanko]." (Perhaps wisely, the executive committee did not mention the participation of three academic philosophers.)

The executive committee also reported an impressive response from industry (especially impressive, given the few months available to obtain it). Several ethics officers or ethics directors of "multinational organizations" had reviewed the code "positively… after sharing it with software engineers in their organizations." Among these organizations were "Lockheed Martin, Northup Grum[m]an, and Digital Equipment."[63] The code had also received support from smaller companies: In fact, the code had already "been adapted by St. Paul Companies as their information systems code of ethics with only minor modifications, such as replacing the expression 'software engineer' with 'IS professional'."

The executive committee then described the "process of continuous review and revision" by which the final Version 4 had reached the Steering Committee. The description seems somewhat garbled. On the one hand, the executive committee wrote, "The final draft of the Code" (that is, Version 3) was published in *Computer* and *CACM* along with a survey. Most items "had greater than 95 percent support with many at 100 percent [with the highest level of opposition being 12%]". Then, having described the overwhelming support for this "final draft", the executive committee continued, "In the light of feedback, the Principles and associated Clauses have been reviewed and modified where necessary." That is, the "final draft" was not the final draft. The executive committee concluded this description of process with the hopeful (but unsupported) statement: "We are confident that if the survey were to be taken again, all Clauses would receive an approval rating of over 95%."

After a brief paragraph demonstrating academic support by "several requests to have the Code included in computer ethics and software engineering textbooks" and other requests to develop cases based on the code, the executive committee reached its

triumphant last paragraph.[64] The code had "already proven to be a dynamic and useful document", a "model" within at least one company; both the ACM and IEEE-CS had "already recognized that education about their codes of ethics is an important issue"; and several members of SEEPP are members of committees charged with educating members "about their codes of ethics". To support "education about the Code", SEEPP planned "to maintain [the code]" on two websites, one at Gotterbarn's SEERI and the other at Rogerson's CCSR. SEEPP would "continue to gather input from the community about the Code and develop and share examples of using [it]." Companies had "already donated case studies related to the Clauses of the Code." It would therefore be good for "the profession if both societies endorsed this effort and approved the Code."

SEEPP's executive committee neglected to mention that it had accomplished all this while delivering the code six days *ahead* of the schedule Gotterbarn had proposed ten months before. The executive committee was also unable to report three more pieces of good news Gotterbarn would receive just before Christmas (December 22). One was an email from a sales manager at Hitachi America's New York City office, a late ballot. The sender had voted in support of every provision, but for several provisions seemed not to have been satisfied with "SF" (strongly favorable) as a way to express accurately how strongly he favored the provision. So, every now and then there was an "x" far to the left of "SF". His only comment was that he would also fax his response.[65]

December 22 seemed to be a day for catching up on correspondence. Earlier in the morning, Gotterbarn had received an email from Bangalore. The author, who worked for Siemens Information Systems (India), wanted to know where he could "get a soft copy of the draft SE code of ethics V2.1 that you published in SEN 22 (July 1997)". Even "plain text" would do. He wanted to make it available online in his software development center for wide circulation."[66] Why had it taken so long for Version 2.1 to reach India—or so long for India to respond? Whatever the answer to that question, Gotterbarn could now add Asia to the continents participating in drafting the code.

Then, after dark, Gotterbarn received another email from Siemens, this one from Germany. Two emails from Siemens in one day? Could that be mere coincidence? The email did not answer that question. It was (like the email from Hitachi) a ballot. But the German had not bothered to print out the long columns because "I (strongly) favor the clauses put forward. I think, the code could be voted on [in] its present form [Version 3.0]." He nonetheless had several suggestions. One was that "clauses 1.04, 1.07, 1.08, 1.09, 1.12, and 1.14 should be inspected more closely".[67] They might be assembled as "an approved collection of methodologies and (possibly augmented) standards." The other comment concerned 6.10 ("Obey all laws governing their work…").[68] The German thought this clause needed clarification. "Is censorship consistent with public welfare?" He wondered whether the clause could be dropped altogether: "If it's illegal, let the government's lawyers worry about it." He even wondered whether what was meant was, "You may disobey the law, if obedience is inconsistent …"

Gotterbarn does not seem to have passed these three emails on to Miller and Rogerson. Instead, he wrote the two (on Christmas day) with a problem. But, before stating it, he described some of his recent correspondence. From the description, we can tell some have been lost. "People are," Gotterbarn began, "still sending in votes on 3.0." One of these, a "vp in Chicago", had passed out the survey to thirteen of his subordinates. "He should," Gotterbarn thought, "be upset by the results—6 of the 13 strongly oppose

'being responsible for your work'!!!!!" Gotterbarn also seems to have responded to the correspondent in India. He was (Gotterbarn reported) an "Information Systems Manager" who had read about the Code "in the July issue of SIGSOFT Newsletter". He wanted to distribute the code "to all of his employees :-)". The only version now available ["4.0"] was on the SEERI website. That, however, was not the problem he wanted help with. The problem was "the long, carefully worked out comments I get by email, some as long as 5 typed pages,—to which we [the executive committee] replied 'we will take these into account in the next version early next year'." Gotterbarn wondered whether he should instead "ignore the emails, or [answer simply] 'your comments will be taken into consideration'."[69]

Miller responded on December 27 (with a copy to Rogerson). It was (he thought) important that "the people do get a response".[70] He wanted them to feel part of the process. But any response could be no more than a response "for the three of us, since we have the authority to speak only for ourselves." Miller suggested the response make four points: First, Version 4.0 (in its final form) is now before the IEEE and ACM governing bodies and is no longer subject to change. (Correspondents should be sent a copy.) Second, "we" should "tell people (and commit ourselves to the project) that we will begin work on version 5.0 in 1998", taking into account "all the feedback we get between now and Thanksgiving 1998." Third, Miller also wanted to write "a proposal (to ACM, IEEE, NSF,…) for us three to get together again in Dec 1998." Last, after that December meeting, they would "start [suggesting] the changes to the powers that be." Apparently, after a week of vacation, Miller was willing to go on a little longer "trying to please everyone over and over."

Rogerson responded two days later (December 29)—with a copy to Miller. Though giving no indication that he had seen Miller's message, he too suggested that Gotterbarn "send an email explaining that we are now in Version 4 and that their comments will be fed into the ongoing maintenance and development of the code and its associated accessories (teaching packs and so on)." Apparently, Rogerson did not anticipate Version 5, but he was sure that the executive committee "must keep all this feedback and eventually write up the findings/outcomes in a paper for CACM or ???"[71]

## 10.8 What next?

That was a good ending to 1997. Would 1998 be a happy new year? The executive committee could not know. They did not even know whether SEEPP still existed or had, upon delivery of Version 4, automatically dissolved into so many individuals again. Indeed, they did not even know whether the Steering Committee still existed. It had, after all, only five months ago threatened to go out of existence at the end of 1997. It gave no indication of life. The transmittal of Version 4 had not been acknowledged. There had not even been a "thank you" from Cabrara or Frailey. If the Steering Committee no longer existed, who would guide the code through ACM and IEEE-CS approval? New problems to replace the old.

10. Appendix:

# *SOFTWARE ENGINEERING CODE OF ETHICS AND PROFESSIONAL PRACTICE*

(version 4) as recommended by the
IEEE-CS/ACM Joint Task Force on Software Engineering Ethics and Professional Practices

## Short Version

## PREAMBLE

The short version of the code gives the aspirational elements at a high level of abstraction; the clauses that are included in the full version give examples and details of how these aspirations change the way we act as software engineering professionals. Without the aspirations, the details can become legalistic and tedious; without the details, the aspirations can become high sounding but empty; together, the aspirations and the details form a cohesive code.

Software engineers shall commit themselves to making the analysis, specification, design, development, testing and maintenance of software a beneficial and respected profession. In accordance with their commitment to the health, safety and welfare of the public, software engineers shall adhere to the following **Eight Principles**:

**1 PUBLIC** - Software engineers shall always act consistent with the public interest.

**2 CLIENT OR EMPLOYER** - Software engineers shall act in the best interests of their client or employer consistent with the public interest.

**3 PRODUCT** - Software engineers shall ensure that their products and related modifications are of the highest professional standards possible.

**4 JUDGEMENT** - Software engineers shall maintain integrity and independence in their professional judgement.

**5 MANAGEMENT** - Software engineering managers and leaders shall subscribe to and promote an ethical approach to the administration of software development.

**6 PROFESSION** - Software engineers shall advance the integrity and reputation of the profession consistent with the public interest.

**7 COLLEAGUES** - Software engineers shall be fair to and supportive of their colleagues.

**8 SELF** - Software engineers shall participate in lifelong learning regarding the practice of their profession.

# *SOFTWARE ENGINEERING CODE OF ETHICS AND PROFESSIONAL PRACTICE*

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

## Full Version

## PREAMBLE

Computers have a central and growing role in commerce, industry, government, medicine, education, entertainment and society at large. Those who contribute, by direct participation or by teaching, to the analysis, specification, design, development, maintenance and testing of software systems have significant opportunities to do good or cause harm, to enable others to do good or cause harm, or to influence 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 software engineering a beneficial and respected profession. In accordance with that commitment, software engineers shall adhere to the following Code of Ethics and Professional Practice.

The Code contains eight Principles related to the behavior of and decisions made by professional software engineers, including practitioners, educators, managers, supervisors and policy makers, as well as trainees and students of the profession. The Principles identify the ethically responsible relationships in which individuals, groups, and organizations participate and the primary obligations within these relationships. The Clauses of each Principle are illustrations of some of the obligations included in these relationships. These obligations are founded in the software engineers' humanity, special care owed to people affected by their work, and the unique elements of the practice of software engineering. The Code prescribes these as obligations of anyone claiming to be or aspiring to be a software engineer.

It is not intended that the individual parts of the Code 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 be in tension 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 and Professional Practice, given the circumstances.

Ethical tensions can best be addressed by thoughtful consideration of fundamental principles, rather than blind reliance on detailed regulations. These Principles should

influence software engineers to consider broadly who is affected by their work; to examine if they and their colleagues are treating other human beings with due respect; to speculate on how the public, if reasonably well informed, would view their decisions; to analyze how the least empowered will be affected by their decisions; and to consider whether their acts would be judged worthy of the ideal professional working as a software engineer. In all these judgements concern for the health, safety and welfare of the public is primary. That is, the "Public Interest" is central to this Code.

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 software engineers and managers of software engineers who need to take positive action in a specific case by documenting the ethical stance of the profession. The Code provides an ethical foundation to which individuals within teams and the team as a whole can appeal. The Code helps to define those actions which are ethically improper to request of a software engineer or teams of software engineers.

The Code is not simply for adjudicating the nature of questionable acts; it also has an important educational function. Since it expresses the consensus of the profession on ethical issues, it is a means to educate both the public and aspiring professionals about the ethical obligations of all software engineers.

## PRINCIPLES

**Principle 1 PUBLIC** Software engineers shall always act consistent with the public interest. In particular, software engineers shall, as appropriate:

1.01. Accept full responsibility for their own work.
1.02. Moderate the interests of the software engineer, the employer, the client and the users with the public good.
1.03. 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.
1.04. Disclose to appropriate persons or authorities any actual or potential danger to the user, the public, or the environment, that they reasonably believe to be associated with software or related documents.
1.05. Co-operate in efforts to address matters of grave public concern caused by software or related documents.
1.06. Be fair and avoid deception in all statements, particularly public ones, concerning software or related documents.
1.07. Consider issues of physical disabilities, allocation of resources, economic disadvantage and other factors that can diminish access to the benefits of software.
1.08. Volunteer professional skills to good causes and contribute to public education concerning the discipline.

**Principle 2 CLIENT OR EMPLOYER** Software engineers shall act in the best interests of their client or employer consistent with the public interest. In particular, software engineers shall, as appropriate:

2.01. Provide service in their areas of competence, being honest and forthright about any limitations of their experience and education.
2.02. Not knowingly use software that is obtained or retained either illegally or unethically.
2.03. Use the property of a client or employer only in ways properly authorized, and with the client's or employer's knowledge and consent.
2.04. Ensure that any document upon which they rely has been approved, when required, by someone authorized to approve it.
2.05. Keep private any confidential information gained in their professional work, where such confidentiality is consistent with the public interest.
2.06. Identify, document, and report to 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, or otherwise to be problematic.
2.07. Identify, document, and report significant issues of social concern, of which they are aware, in software or related documents to the employer or the client.
2.08. Accept no outside work detrimental to the work they perform for their primary employer.
2.09. Promote no interest adverse to their employer or client, unless a higher ethical concern is being compromised; in that case, inform the employer or another appropriate authority of the ethical concern.

**Principle 3 PRODUCT** Software engineers shall ensure that their products and related modifications are of the highest professional standards possible. In particular, software engineers shall, as appropriate:

3.01. Strive for high quality, acceptable cost and a reasonable schedule, ensuring significant tradeoffs are clear to and accepted by the employer, the client, the user and the public.
3.02. Ensure proper and achievable goals and objectives for any project on which they work or propose.
3.03. Work to identify, define and address ethical, economic, cultural, legal, and environmental issues related to work projects.
3.04. Ensure that they are qualified, by an appropriate combination of education, planned education, and experience, for any project on which they work or propose to work.
3.05. Ensure an appropriate methodology for any project on which they work or propose to work.
3.06. Work to follow industry standards when available that are most appropriate for the task at hand, departing from these only when technically justified.

3.07. Strive to fully understand the specifications for software on which they work.
3.08. Ensure that specifications for software on which they work have been well documented, satisfy the users' requirements, and have the appropriate approvals.
3.09. Ensure realistic estimates of cost, scheduling, personnel, and outcomes on any project on which they work or propose to work and provide a risk assessment of these estimates.
3.10. Ensure adequate testing, debugging, and review of software and related documents on which they work.
3.11. Ensure adequate documentation, including significant problems discovered and solutions adopted, for any project on which they work.
3.12. Work to develop software and related documents that respect the privacy of those who will be affected by that software.
3.13. Be careful to use only accurate data derived by ethical and lawful means, and use only in ways properly authorized.
3.14. Maintain the integrity of data, being sensitive to outdated or flawed occurrences
3.15 Treat software maintenance with the same professionalism as new development.

**Principle 4 JUDGEMENT** Software engineers shall maintain integrity and independence in their professional judgement. In particular, software engineers shall, as appropriate:

4.01. Temper all technical judgements by the need to support and maintain human values.
4.02 Only endorse documents prepared under their supervision or within their areas of competence and with which they are in agreement.
4.03. Maintain professional objectivity with respect to any software or related documents they are asked to evaluate.
4.04. Not engage in deceptive financial practices such as bribery, double billing, or other improper financial practices.
4.05. Disclose to all concerned parties those conflicts of interest that cannot reasonably be avoided or escaped.
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.

**Principle 5 MANAGEMENT** Software engineering managers and leaders shall subscribe to and promote an ethical approach to the administration of software development. In particular, those managing or leading software engineers shall, as appropriate:

5.01 Ensure good management for any project on which they work, including effective procedures for promotion of quality and reduction of risk.
5.02. Ensure that software engineers are informed of standards before being held to them.

5.03. Ensure that software engineers know the employer's policies and procedures for protecting passwords, files, and other confidential information.
5.04. Assign work only after taking into account appropriate contributions of education and experience tempered with a desire to further that education and experience.
5.05. 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.
5.06. Attract potential software engineers only by full and accurate description of the conditions of employment.
5.07. Offer fair and just remuneration.
5.08. Not unjustly prevent someone from taking a better position for which that person is suitably qualified.
5.09. Ensure that there is a fair agreement concerning ownership of any software, processes, research, writing, or other intellectual property to which a software engineer has contributed.
5.10. Provide for due process in hearing charges of violation of an employer's policy or of this Code.
5.11. Not ask a software engineer to do anything inconsistent with this Code.
5.12. Not punish anyone for expressing ethical concerns about a project.

**Principle 6 PROFESSION** Software engineers shall advance the integrity and reputation of the profession consistent with the public interest. In particular, software engineers shall, as appropriate:

6.01. Help develop an organizational environment favorable to acting ethically.
6.02. Promote public knowledge of software engineering.
6.03. Extend software engineering knowledge by appropriate participation in professional organizations, meetings and publications.
6.04. Support, as members of a profession, other software engineers striving to follow this Code.
6.05. Not promote their own interest at the expense of the profession.
6.06. Obey all laws governing their work, unless, in exceptional circumstances, such compliance is inconsistent with the public interest.
6.07. Be accurate in stating the characteristics of software on which they work, avoiding not only false claims but also claims that might reasonably be supposed to be deceptive, misleading, or doubtful.
6.08. Take responsibility for detecting, correcting, and reporting errors in software and associated documents on which they work.
6.09. Ensure that clients, employers, and supervisors know of the software engineer's commitment to this Code of ethics, and the subsequent ramifications of such commitment.
6.10. Avoid associations with disreputable businesses and organizations.
6.11. Consider that violations of this Code are inconsistent with being a professional software engineer.

6.12. Express concerns to the people involved when significant violations of this Code are detected unless this is impossible, ineffective, or dangerous.
6.13. Report significant violations of this Code to appropriate authorities when it is clear that consultation about the problems is impossible, ineffective or dangerous.

**Principle 7 COLLEAGUES** Software engineers shall be fair to and supportive of their colleagues. In particular, software engineers shall, as appropriate:

7.01. Encourage colleagues to adhere to this Code
7.02. Assist colleagues in professional development.
7.03. Credit fully the work of others and refrain from taking undue credit.
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 and other confidential information, and security measures in general.
7.07. Not unfairly intervene in the career of any colleague; however, concern for the employer, the client or public interest may compel software engineers, in good faith, to question the competence of a colleague.
7.08. 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 participate in lifelong learning regarding the practice of their profession. In particular, software engineers shall continually endeavor to:

8.01. Further their knowledge of developments in the analysis, specification, design, development, maintenance 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 produce accurate, informative, and literate documentation.
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. Not influence others to undertake any action which involves a breach of this Code.
8.08. Consider that personal violations of this Code are inconsistent with being a professional software engineer.

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

NOTES

[1] Gotterbarn\Version 3\Comment\Unger-4 (addressed to Gotterbarn only). The "4" after "Unger" indicates that this was the fourth "ballot" in Gotterbarn's tally (and perhaps the fourth to arrive). One of the first three seems to have been Notkin's (no number); LaRue's (below) seems to have come third. Whose was the other? Is one lost? Or was it just a ballot without any comment? Neither the archive nor Gotterbarn can tell.

[2] Unger, who holds a BS (Brooklyn Polytechnic, 1952), MS (MIT, 1953) and ScD (MIT, 1957), all in electrical engineering, had worked at Bell Labs (1957-61) before becoming a professor. As part of a re-organization there, he was asked to head a group to develop a compiler for the first electronic telephone switching system (ESS). He agreed to do that for a limited time. He was in charge of the compiler group for two years. Later he did theoretical work on software (e.g., a program for a syntax analyzer). He is now a professor in the Computer Science Department of Columbia University and also a professor in the Electrical Engineering Department. Unger dislikes the term "computer science": "Science is about understanding nature. Engineering is about making artifacts. Software is an artifact. Computer science is about making computers and software. That's engineering, as far I am concerned." Interview of Unger, November 13, 2002. Though never involved in SEEPP, he had been in contact with Gotterbarn for some time. His name first appears in Gotterbarn\SEEP 1994-96\PCVOL3 (file date 6-8-95) in an unsigned note (apparently sent to Gotterbarn by some third person):

> Steve Unger is on the new IEEE International Ethics Committee and with others we are helping Steve implement a mechanism to help IEEE members (later we hope to extend to others working in EE/CS) in ethics matters. The support of your task force is desired. (We of course need to tell you about it. By CC to Steve I'm asking him to do so.
>
> Apart from that, I would like to be involved in SEEPP in some way. Thanks.

The official title of the committee in question is "IEEE Ethics Committee". The parent organization of IEEE-USA is just IEEE (though most members I know tend to refer to IEEE-USA as "IEEE" and the world-wide parent as "IEEE International"—when they bother to distinguish the two).

[3] Stephen Unger, *Controlling Technology* (John Wiley and Sons: New York, 1994). Among his important articles then were: "Role of Engineering Societies in Controlling Hazardous Technology", *Journal of Professional Issues in Engineering* 112 (July 1986): 151-157; "Would Helping Ethical Professionals Get Professional Societies in Trouble?" *Conference Record – Electro*, 1987 (5. 4. 1-5. 4. 6); and "BART Case: Ethics and the Employed Engineer", *Communications Society* 12 (May 1974): 11-13. He had, of course, also published a great many technical papers.

[4] The reason for the substitution of the 1990 code for the 1987 one is not clear, but it seems to be a product of complex relations between IEEE-USA and its parent organization ("IEEE").

[5] These were the Ethical Guidelines Walter Elden's group was working on (9.1).

[6] The phrase "fair and truthful" was my rendering of the standard language of many engineering codes, "objective and truthful". I had substituted "fair" for "objective" because of post-modern concerns about the possibility of objectivity. Unger's suggestion may be an instance of a tendency all writers notice. Change one word in a well-known phrase and it becomes much easier to change others. Perhaps if I had not substituted "fair" for "objective", Unger would not have suggested the double negative, "avoid deception". On a better day, he might have suggested something more positive, such as "candid" for "truthful". Anyone who writes a code will, I think, soon be surprised at what gets read in or read out. Gotterbarn still thinks that my "fair and truthful" does not "not clearly eliminate the possibility of deception by omission". Gotterbarn Chapter9.cmtd (September 29, 2004). I thought "fair" eliminated deception by omission.

[7] Gotterbarn\Version 3\Survey Comments\LARUE-3.

[8] The six Uncertains were: 1.10 ("respect privacy"), 1.11 ("derive data from legal sourses", 1.13 ("identify ethical, economic…issues"), 2.01 ("disclose dangers"), 6.07 ("remuneration appropriate to qualifications"), and 8.08 ("violations of the code").

[9] The two Opposed were: 2.07 ("not put self-interest") and 3.07 ("refuse to participate").

[10] In Version 1, 2.01 had read: "Disclose to appropriate persons any danger that the software or related documents on which they work may pose to the user, a third party, or the environment."

[11] In Version 1, this clause was 2.06: "Not put self-interest, the interest of an employer, or the interest of a client or customer ahead of the public's interest." Version 3 had dropped "customer" (presumably because it is redundant) and added "the interests of the user" (presumably because the user's interest is not identical with the public interest). I regard this change as an improvement—and clearly not the cause of the opposition of our Omaha respondent.

[12] Perhaps the Omaha respondent was proposing a return to the approach taken in the IEEE's first code of ethics (the 1914 code of ethics of what was then the American Institute of Electrical Engineering). Paragraph A.2 read:

> It is the duty of the engineer to satisfy himself to the best of his ability that the enterprises with which he becomes identified are of legitimate character. If after becoming associated with an enterprise he finds it to be of questionable character, he should sever his connection with it as soon as practicable.

Engineers seem eventually to have decided that such a pre-qualification clause did not provide enough protection for the public.

[13] The respondent from Omaha objects to 3.07 that "'refuse' is too strong" and to 6.07 that he does not see how "appropriate remuneration [is to be] determined".

[14] Gotterbarn\Version 3\Survey Comments\CARTMEL. The author of this email was then the principal in Cartmell Technologies of Kingston, New York, a consulting firm advising on assistive technologies. http://www.ulster.net/~dcartm.

[15] Clause 1.07 (Version 3.1) read: "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."

[16] Gotterbarn\Version 3\Survey Comments\TAYLOR. This respondent has a BS (1989), an MS (1991), and a Ph.D. (1994), all in Computer Science and all from University of North Carolina-Chapel Hill. His dissertation was titled "The Nanomanipulator: A Virtual-Reality Interface to a Scanning Tunneling Microscope". Since 1994, as an NSF CISE Postdoctoral Research Associate (and with additional obtained funding), he has directed the nanoManipulator project. Research interests include virtual environments, man-machine interaction, scientific visualization and distributed systems. http://www.cs.unc.edu/ ~taylorr/taylorr_CV.html.

[17] Email (Gotterbarn to Davis), February 22, 2004.

[18] November 13, 1997.

[19] "Nov 3 The steering committee has asked me to get comments from industry". Gotterbarn\History of SE Code\History expanded.

[20] E971118 and E971120.

[21] Gotterbarn\Version 3\Survey Comments\LANE40. The number "40" seems here to indicate the place of Lane's response in some final reckoning, not its place in the temporal order of responses. So, for example, LINDLEY29 arrived two weeks after LANE40.

[22] The TI respondent must have been thinking of large systems designed for sophisticated clients, not shrink-wrapped software sold to ordinary consumers.

[23] Clause 1.12 (Version 3.1) read: "Whenever appropriate, delete outdated or flawed data". Clause 2.02 read: "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."

[24] By December 1, 1997, there were at least 65 ballots. A few others seem to have come in too late to be counted. I say "At least 65" because that is the highest number of votes on any line of the table saved in Gotterbarn\Version 3\V3 (a file saved on 12-3-97). Is 65 a good response or a bad? In proportion to the hundreds of thousands of software engineers who could have responded, 65 seems almost no response at all. But, given the

quality of the responses, and the small number of respondents in any step of the process so far, 65 looks pretty good.

[25] Email (Gotterbarn to Davis), February 22, 2004 (departmental support). Gotterbarn\Version 3\Survey Comments\ALLCMNTS. This is identical with: Survey Comments\CMTS20NOV.

[26] This is a good idea for dealing with a persistent complaint resting on an undefended prejudice. But it is not as radical or as ACMish as Gotterbarn makes it sound. On the one hand, even Version 1's Rules were designed so that (minus the "In particular…") they could stand alone to form a short code. After all, I had before one major engineering codes (ABET's) that had a short and a long version related in something like this way. The executive committee did not need to change the wording of any Principle. On the other hand, the ACM, having originally printed its document with the code of ethics first and the guidelines after (with each set of the guidelines preceded by the appropriate provision of the code) was by 1997 printing just the guidelines. ACM seems to have treated the short version as a mere inconvenience necessary for approval. By 1997, what Gotterbarn was proposing was not (or, at least, no longer) the "ACM model".

[27] IIT Archive, 1997, New Folder (November 21, 1997).

[28] IIT Archive, 1997, New Folder (November 21, 1997).

[29] Gotterbarn\SEEP1994-65\OPGUIDE.

[30] 4.3.5.c ("This category is provided to allow for ballot returns from members who do not wish to review documents because of conflict of interest, lack of expertise, or other reasons. A reason shall be given for this vote; otherwise, the ballot shall be classified as 'no response.'")

[31] Gotterbarn\Version 3\Survey comments\STPAUL.

[32] Gotterbarn, email (March 2, 2004) added that smaller companies seldom have such adoption problems. His Google search using "Software Engineering Code of Ethics" even revealed small companies that use adoption of the code in their advertising.

[33] Gotterbarn\Version 3\ALLCMNTS (file date 12/3/1997). This seems to be the same as: Gotterbarn\Version 3\Survey Comments\ALLCMNTS (file date 12/1/1997). The earlier document is actually one third longer, but there are several anomalies (such as four pages in which words are printed one letter per line) that make comparing length harder.

[34] When there was no vote at all on a particular clause, the blank was ignored in all calculations.

[35] Gotterbarn\Version 3\VER3mod.

[36] The twenty clauses 3b set in bold for inspection were: 1.04, 1.05, 1.06, 1.08, 1.12, 1.15, 2.02, 2.05, 2.07, 2.08, 3.05, 4.02, 4.06, 4.09, 6.03, 6.05, 6.07, 6.13, 7.02, and 7.08.

[37] So, for example, the list of names at the end of the code ("members") now include Weil and me, although there is no indication that it has been changed.

[38] Gotterbarn, email (March 3, 2004).

[39] These comments are now lost. Gotterbarn recalls that "some commented briefly on several of the clauses, …describing why they had voted negative or positive, advocating that a clause be made even stronger, etc." But, from the summary of comments, it seems to me that the faxed ballots could not have contained much beyond the vote itself—or there would be comments quoted in the summary that do not correspond to emails saved in Gotterbarn's archive. I have been unable to identify any. Gotterbarn, email (March 4, 2004) and (March 8, 2004 attachment).

[40] Email (Miller to Davis), March 15, 2004.

[41] Gotterbarn, email (March 4, 2004). Miller, email (March 15, 2004).

[42] Gotterbarn, email (March 4, 2004).

[43] Gotterbarn, email (March 3, 2004). Compare Miller (email, March 15, 2004):

> Many academics love to debate (myself included), but often that can lead to acrimony. In this case, I found it exhilarating to be in an intense, intellectual exchange while knowing that each of us was working towards the same goal—a better code. I think we trusted and respected each other enough to say exactly what we thought, and I never felt attacked or even annoyed. We could get exasperated sometimes, but I remember us being in good humor most of the time. (Don and Simon "ganged up" on me once because I didn't like a word or spelling, and they assured me it was good English, from the United Kingdom, and that I was being a contrary Yank, or something like that.)

[44] Gotterbarn on this point (Email, March 3, 2004):

[45] While 7.08, with one of the weakest endorsements (85%), was marked for revision or deletion, 2.03 was not. With almost 97% in favor of it, it should have been immune to change, much less deletion. At the least, such deletions should have been noted. Gotterbarn\Version 3\V3 Survey Votes.

[46] Version 4 was 2706 words compared to Version 3's 3407 (and Version 1's 1517). It is interesting to note that Version 4 is about the same length as Version 2.1 (2759). It is 3.0 (3407 words) that is the outlier among the executive committee's codes—and, indeed, among major codes from engineering and computer science. The complete ACM code is

2837 words; the NSPE's, 2218; and the ABET Guidelines, 2666. Did Gotterbarn realize that Version 3 was unusually long? He certainly knew that Version 4 was much shorter. See Gotterbarn\Version 4\MODS ("A brief summary of the general direction of the modifications on version 3 to make 4"), a rough draft never completed (file date 12/16/1997): "The Code was reduced in size by 30%...."—There seem to be at least three ways to calculate the reduction from 3.0 to 4.0 so that it is 30% (as opposed to the 25% I calculated): 1) treat Version 4 rather than Version 3 as the base (though that would yield a 33% reduction); 2) treat only the first digit of the percentage as a significant number (and rounding up to 30% from 25.6%); or 3) combining 1 and 2 (rounding down from 33%).

[47] Gotterbarn\Version 4\S12), December 12, 1997.

[48] Why had I responded to this call for comment when I had not responded to others Gotterbarn had issued? In part, I responded because I could easily do so. The request had arrived when classes were over, all papers graded, and the final exam ready but not yet given. I had a relatively clean desk. (Had the request come a week earlier, I would not have been able to respond in time, since I was then out of the country—and offline.) But, in part too, I responded because I thought it was time to give some advice on drafting. I was hinting that the executive committee would do well to choose me as the editor I had suggested. Their choosing me for the job did not seem as much of a stretch as it would have six months earlier. By then, Gotterbarn knew much of the story of Version 1 because I had asked him to comment on a first draft of a little paper eventually published as: Michael Davis, "Writing a Code of Ethics by E-Mail: Adventures with Software Engineers", *Science Communication* 21 (June 2000): 392-405. (I thanked Gotterbarn for his comments in a postscript to the December 9 fax.)

[49] I also suggested that 4.04 should not begin "Do not engage" because the "shall" in the introductory "In particular" phrase above controls: "one modal is enough." (The final Version 4 corrected this error.) From my perspective, and perhaps from the executive committee's, we were involved with "debugging" a pretty "buggy" code.

[50] I also took exception to the language of this Principle: "'within confines of'. Dead metaphor. Presumably, it means the same as the shorter (and more common) 'consistent with'." The final Version 4 was revised accordingly.

[51] Gotterbarn probably did not treat my comments in this way because a three-page fax was not easily forwarded to the list. He would either have had to retype fax or scan it (and, in 1997, scanners were still not nearly as common, reliable, or easy to use as they would be even two years later). Gotterbarn must, however, have passed my comments on to Rogerson and Miller (in some form or other). Miller and Rogerson (in their responses to Fairweather's comments below) each refer to my comments once (though both call me "M Davies", indicating a single, but mangled, source). The references are to different parts of my comments.

[52] Gotterbarn\Version 4\KEITH.

[53] I had a different objection to Principle 3, specifically to its "products and related modifications": "Modifications are generally modifications of something. If of products, they are products. If of something else, what? Frankly, 'modifications' just strikes me as confusing." Even Version 5.2 retained this unfortunate expression.

[54] In the next version of the code (December 18, 1997), the short version had two paragraphs instead of one, the paragraph coming first doing as Fairweather had suggested. Rogerson's okay was not always so potent, however. For example, having noted the change from "employee" to "software engineer" in 5.02 and other clauses under Principle 5, Fairweather pointed out that a software engineer "may be leading a team that includes people other than software engineers (for example, administrative support)." The good software engineer-manager "should not punish +any+ employee for expressing ethical concerns about a project, or ask any employee to do anything inconsistent with this Code, [and so on]." Rogerson seemed to concede the point, "Interesting—suggest we change 'a software engineer' to 'anyone'." Miller did not object. Yet, the next version—and all its successors—preserved "software engineer". Apparently, sometimes Gotterbarn's vote was all that mattered.

[55] "Without the aspirations, the details can become legalistic and tedious; without the details, the aspirations can become high sounding but empty; together, the aspirations and the details form a cohesive code." Philosophers may wonder at the similarity between Miller's language and a famous aphorism in Kant's *Critique of Pure Reason*: "Thoughts without contents are empty, intuitions without concepts are blind."

[56] "Gotterbarn\History of SE code\History expanded" reports that the web posting occurred two days later: "Dec 18 version 4 place on SEERI web site announcing that it was recommended to the two societies."

[57] The expression "being sent" is, perhaps, a bit misleading. The suggestion is that the document is even now on its way to the Steering Committee. In fact, from dating on the files of supporting documents, it seems likely that Gotterbarn did not actually email the "final version" to the Steering Committee until the next day, December 16 (after his grades had been turned in). December 16 is also the date given in Gotterbarn's own chronology: Gotterbarn\History of SE code\history expanded ("Version 4 to both societies for approval- December 16 1997"). A related technicality: By Gotterbarn's usual measure, the version sent the Steering Committee (what the email to the task force called "Version 4.DONE") should have been "Version 4.1", since its immediate predecessor was Version 4.0 (December 7, 1997) and that Version 4 differed from this Version 4 in many ways small and large. In fact, "Version 4. DONE" (December 15, 1997) would thereafter be Version 4. This *is* confusing. So, wherever there is a risk of confusion, I will date the version of 4 in question or otherwise make clear which I intend. (Gotterbarn explained his notation this way: "Even though there is a big difference between the 4 of December 7 and the 4 of December 15, we did not bother to assign different version numbers to the paper copies because we did not consider the submission

to the task force to be a release, only a step in drafting." Email, Gotterbarn to Davis, March 8, 2004.)

[58] Because Rogerson was writing from England, the timestamp for this message is 2:30 PM (that is, 9:30 AM CST).

[59] Late that afternoon (December 16, 1997), Ron Prinzivalli wrote the listserv, "What is this?" Was he asking about Version 4.DONE or the parties in Rotterdam and Washington? We cannot tell. No one seems to have responded to his cry for clarification—and he was already dead when I called for an interview (September 2003).

[60] December 17, 1997.

[61] Gotterbarn\Steering Committee\Recommendation of 4\REC (December 16, 1997). The letter begins (after the date) with the salutation: "To the Joint IEEE-CS/ACM Steering Committee on the Professionalization of Software Engineering and Executive Committees of the ACM and the IEEE-CS." Because the file is not itself an email, we cannot tell to whom this message was sent or when. But it seems reasonable to suppose that Gotterbarn would have sent it to Cabrera, with a copy to Frailey, as he regularly did when dealing with the Steering Committee, and that he would have sent the letter to the other Committee members through the presiding officer (rather than directly). The only confusion seems to concern to whom the recommendation was sent. "Gotterbarn\History of SE code\History expanded" mentions only the two societies (omitting the Steering Committee). This seems a slip. The Steering Committee's chair and co-chair never denied receiving the December 16, 1997, letter of transmittal. The importance of these details will become obvious in the next chapter.

[62] For more detail, including the exact working group involved and what it reported, see 8.7 (and endnote).

[63] The archives contain no record of this correspondence.

[64] These requests are not in the archive.

[65] Gotterbarn\Version 3\Feedback on 3\Wentworth (December 22, 1997).

[66] Gotterbarn\Version 3\Survey comments\Siemenscommnt.

[67] The language of these clauses does not matter here. For the exact wording of these clauses, see 9. Appendix.

[68] Version 3.0's 6.10 ("Obey all laws governing their work, insofar as such obedience is consistent with the public health, safety, and welfare ") had become Version 4's 6.04 ("Obey all laws governing their work, unless, in exceptional circumstances, such compliance is inconsistent with the public interest"). Would this 6.04 have satisfied the German? What of the ancestor of all these, Version 1's 5.10 ("Obey all laws governing

their work, insofar as such obedience is consistent with the public health, safety, and welfare")?

[69] The original of this email has not survived, but we have it complete both in Miller's response (Gotterbarn\Version 3\ Survey Comments\AKM) and Rogerson's (Gotterbarn\Version 3\Survey Comments\ASI).

[70] Gotterbarn\Version 3\ Survey Comments\AKM.

[71] Gotterbarn\Version 3\Survey Comments\ASI.

# Chapter 11: The Long Process of Approval, 1998

"Whenever you set out to do something, something else must be done first."
—Murphy's Law #12

## 11.1 An unpleasant surprise

After all the activity of 1997, Gotterbarn enjoyed the relative calm of the first few weeks of 1998. Only a few Code-related emails, like stragglers after a great battle, drifted in from time to time. The first arrived on January 7. Its author, a sixth-year student in computer science at the Technical University of Eindhoven, had a page and a half of "opinions and criticisms" concerning Version 3 (9. Appendix). While he liked most of the Code, he thought it "geared towards software engineers working in a (large) company." As an "independent software engineer", he "personally" disliked "this point of view". Since most codes of engineering ethics have been criticized for not paying enough attention to engineers working in large organizations (large organizations being where most engineers work), the Dutch student had, however unintentionally, complimented the Gotterbarn.

The Dutch student also thought the Code felt "very US-oriented". Several "political[ly] correct terms" obscured "the meaning for us foreigners". Had Version 4 taken care of that problem (10. Appendix)? For example, the old "diversity" clause (renumbered 1.07) now read: "Consider issues of physical disabilities, allocation of resources, economic disadvantage and other factors that can diminish access to software." No "politically correct terms" there. Perhaps Version 4 had taken care of another problem as well.[1] The Dutch student was concerned that some clauses "may be in violation of local laws". The only example he gave was "5.05" ("Develop a fair agreement concerning ownership of any software, processes, research, writing, or other intellectual property to which an employee has contributed"). Version 4 had dropped that clause; perhaps it had done the same with others the Dutch student might have mentioned.[2] This email from the Netherlands seemed to endorse what the executive committee had done in Baltimore.

Much the same was true of a much longer (six page) email, equally thoughtful, that arrived the next day from Switzerland. Its author, George Sigut, held a position at the Federal Institute of Technology in Zurich much like that Manny Norman held at Eastern Michigan. In charge of Computer Services for the Institute, he was a software engineer practicing within an academic (research) environment.[3] Sigut proposed: "1 Changes in the form [;] 2 Changes in the Principles [; and] 3 Other changes". One of the changes in form, though different from what Gotterbarn, Miller, and Rogerson had made in December, was surprisingly similar. Sigut proposed that, instead of Version 3's two-part division into Preamble and Principles (with Clauses), the Code should have a three-part division: Preamble; Principles; and Clauses. He wanted the reader to be able "to consider the Principles on their own [without a major effort]". Version 4's "short version" made that possible. Like some other critics, Sigut also wanted to distinguish clauses according to the three categories described in Version 3's Preamble (Aspire, Expect, and Demand). Gotterbarn, Miller, and Rogerson had considered doing that too, but had instead abandoned explicit mention of the categories to avoid confusion. Perhaps Sigut would

have agreed that deletion of the distinction was an equally good way to resolve the problem he had identified. Why not?

Sigut had given the Preamble considerable attention. His suggestions for changes constitute about a quarter of the entire six-page email. Their general drift is to sharpen the distinction between Principles and Clauses (a distinction fundamental to the design of the Code yet a source of misunderstanding ever since Version 1). For example, his "Point 5" is to amend the fifth paragraph by deleting "Principles and" from the sentence, "The list of Principles and Clauses is not exhaustive." The reason for this change is (he suggested) obvious: "Principles are exhaustive by definition". He was, in part, suggesting a reversion to the language of Version 1 ("no set of subsidiary clauses exhausts the general rule [Principle]"). But he also seemed to be making a larger point about the Principles, something Version 1 did not. The Principles defined the special responsibilities of software engineers (what moral obligations they have beyond what everyone else has). The Principles might, if badly designed, leave something out that should have been included. But what was left out would be something that *should* be a special responsibility, not something that *is*. That is the sense in which the Principles are exhaustive "by definition". They were supposed to be exhaustive (whether or not they in fact were). The clauses were not like that. They were just instances of the Principles. There could always be more instances. They were *not* supposed to be exhaustive.[4]

Sigut also picked out for revision many of the clauses Gotterbarn, Miller, and Rogerson had already deleted, for example, old 6.07 ("Only accept remuneration appropriate to professional qualifications or experience"). Here and there, though, Sigut identified a problem Gotterbarn, Miller, and Rogerson had not. For example, Sigut says of old 7.06 (also new 7.06) that the wording might be improved. It then read: "Assist colleagues in being fully aware of current standard work practices including policies and procedures for protecting passwords, files and other confidential information, and security measures in general." Would it not, Sigut asked, read better if rewritten to say: "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"? Apparently, Sigut thought "and security measures in general" looked tacked on to a well-ordered list. He suggested inserting it in the list (without the "and"); there would then be only one (longer) list and only one "and" instead of two.[5]

Gotterbarn now had a set response for late comments on Version 3. It began, "Dear Enlightened Person" (until the individual's name was inserted). After thanking the enlightened ones for "your thoughtful response", it informed them, "We have already factored in numerous responses to version 3 of the code and have produced version 4.0." Version 4 (a copy of which was printed below the letter) "is now up before the IEEE-CS/ACM Steering Committee." Nonetheless, "[your] comments will be fed into the ongoing maintenance and development of the code and its associated accessories (teaching packs and so on)." The results of "this effort" are to be available "early this year" at the ACM and IEEE websites (and at the website for Rogerson's center). The names of the three members of SEEPP's executive committee appear below the final "Sincerely".

Gotterbarn replied on the same day he received Sigut's email (January 8). Though he might have sent Sigut the standard letter, apparently he thought Sigut deserved better. Addressing him as "Dear George", Gotterbarn inserted two sentences in the middle paragraph of the standard reply: "The three level distinction in the preamble drew endless debate about which

clause went at which level. We removed the distinction from the Preamble and (in general) put the more aspirational elements at the top."[6]

A few days later (January 11), another correspondent proposed a new principle or clause (apparently designed to cut across existing Principles):

> Perform factually-based work analyses and estimations to define validatable and verifiable systems requirements, then only budget adequate resources for hard requirements or seek approvals for reduced scopes of work based on hard budgets prior to specifying, accepting and assigning the work.

The author of this disastrous sentence had a legitimate problem he was trying to solve. Many projects "fail because they have a fatally flawed set of requirements or a contradictory mission and budget." They are wasteful because of "some preliminary character flaws of the planners and upper managers". The software engineer should therefore be responsible for evaluating the initial specifications, revising or rejecting them as appropriate, not just accepting them in the way a "servant" would. Who would disagree? But why had the author of this email not noticed that Version 3 (like Version 4) had in fact dealt with the problem? 0ld 1.04 (like new 3.02) required software engineers to: "Ensure proper and achievable goals and objectives for any project on which they work or propose."[7] Had the author of this email read the Code? Perhaps not. There was nothing in the email to show that he had.

This commentator plainly deserved nothing more than the standard letter (with perhaps a reference to the clause that already did what he suggested). But another plainly deserved a response even more personalized that the one Gotterbarn sent Sigut. That commentator had sent an email only a half page long, but its author, Stuart Zweben, an ACM-appointed member of the original Steering Committee (and retiring ACM president). Gotterbarn agreed that Zweben had "identified two of the sore spots in version 3.0." But, in "early December we did a significant revision of the [Code]." Zweben had not liked the wording of old 3.07 ("Refuse to participate in decisions of a governmental or professional body…") or old 2.07 ("Not put self-interest, the interest of an employer…"). Gotterbarn quoted the wording of the corresponding provisions of Version 4 (4.06 and 1.02), suggesting that the new wording had taken care of the problems Zweben identified. Clause 4.06 allowed participation but only after disclosure of any conflict of interest; 1.02 required only that the software engineer "moderate" the interests of self, employer, and so on with the public good. Gotterbarn concluded by noting that he had "tried to address your other question in my 19 January letter to the Steering Committee". He also informed Zweben that "Version 4.0 is available at www.acm.org/serving." Gotterbarn seems to have sent this response to Zweben on January 19.[8]

Zweben's email implicitly told Gotterbarn something that worried him. When he sent Version 4 to the Steering Committee (that is, to Cabrera and Frailey) on December 16, Gotterbarn was anticipating an adoption process rather like that by which the ACM had adopted its code of ethics. The Steering Committee would promptly approve sending Version 4 to the two societies. The Board of Governors of the IEEE-CS would meet in February, consider Version 4, and approve it (perhaps with a few amendments); the ACM's Executive Committee would do the same about the same time. Gotterbarn might have to go before the IEEE-CS Board or the ACM Executive Committee to explain this or that; he might have to meet privately with individual members of one or the other body; but it would all be over by March. By the first day of Spring,

1998, Version 4, or something very like it, would be the official Software Engineering Code of Ethics and Professional Practice; Gotterbarn could turn his full attention to its dissemination—and to teaching, to his Institute, and to his other projects. But that schedule required the Steering Committee to vote on the Code soon after receiving it. For the last month, Gotterbarn had been assuming that, though the Steering Committee had not responded to him, it was at least working on his recommendation, the holiday season explaining any delay in taking the official vote. What Zweben's comments on Version *3* told him was that the Steering Committee was not working on the recommendation of Version 4. Indeed, it told him something worse: Cabrera and Frailey had not, during the intervening month, even told the rest of the Steering Committee that Version 4 existed. Why not? Why had Frailey rushed SEEPP to complete work before the end of the year if he was not even going to pass the Code on to the Steering Committee's members in December?

To find out, Gotterbarn emailed the entire Steering Committee, not just its chair and vice chair, a letter similar in substance (but not organization) to the letter by which Gotterbarn had originally transmitted Version 4.[9] The first paragraph announced that SEEPP had sent the Steering Committee a recommendation in December and had "not heard any response" since.[10] The second paragraph worried about timing: "I believe the Board of Governors of the IEEE-CS and the Executive Committee of the ACM have meetings scheduled in February and I would be willing to represent the SEEPP task force at these meetings." Having the Code approved at those meetings would be "salubrious [because] there are already several events scheduled for February and March which include the Code."[11] Gotterbarn "fired [this] second letter to the whole committee 20 January."[12]

By January 27, Gotterbarn had learned enough to write Miller with details "[just] to keep you up on the fun".[13] Both Zweben and (Dennis) Frailey had responded to Gotterbarn's "second letter". Zweben was "sorry [but] the ACM EC met last week and the next meeting in the US is in May." In other words, the Steering Committee's failure to act quickly had put the ACM schedule for approval back at least three months. That news was distressing enough, but Frailey's was worse. Frailey was, Gotterbarn told Miller, "our only real contact with the committee". Cabrera, having become involved with a major project at Microsoft soon after becoming chair, had become increasingly distant from the Committee's work (much as Gotterbarn's co-chair, Melford, had two years before). Though only vice-chair, Frailey had been trying to hold the Steering Committee together. He was not doing well. He "did not know the status of the steering committee". The quotation marks are Gotterbarn's around Frailey's exact words. Gotterbarn reported replying: "co-chair [sic] did not know status of steering committee????!!!!!" Frailey explained that "no one on the committee is responding to his emails." (The Committee had long since ceased having face-to-face meetings; and, Gotterbarn concluded, even telephone conferences had ceased many months ago.) Frailey was almost ready to "take unilateral action". Here Gotterbarn added his own smile symbol and continued, "We have some experience with this form of in-action from the Steering Committee." So, Gotterbarn was wondering whether he should suggest that Frailey email the Committee saying that, there being no objection, he would forward the recommendation to the two societies by Valentines Day, "the first completed deliverable from the Joint Steering Committee." Gotterbarn would, in other words, suggest "our mode of operation—if he hears no dissent from his committee, Dennis should forward the Code and recommendation to both societies." Gotterbarn ended this chuckling note with a serious

question: “What do you think of this as a recommendation to Dennis? Any suggestion on how he should frame his proposal to the Steering Committee to minimize potential dissent?”

Gotterbarn must have reached Miller on one of his dark days. He responded that Gotterbarn could suggest the silence-means-consent strategy “but I doubt that it would be accepted. I think they WANT to do a pocket veto.” Miller did not explain who “they” were (Cabrera and Frailey, some other members of the Steering Committee, or the Steering Committee as a whole) or why “they” would want to do a pocket veto. Whoever “they” were, Miller thought the better strategy was to go around them: “request, in strong terms, that you or the whole exec. Committee be invited to the May meeting and that we get some guaranteed time on the agenda.” It may be the time “to meet this thing head on instead of out-maneuvering them.” Gotterbarn took Miller’s advice, though he waited till April to do it. Meanwhile he tried to find out more about the status of the Steering Committee, the likely reception the Code would receive at the May meeting of the ACM Executive Council if the Steering Committee remained dormant, and the likely reception at the May meeting of IEEE-CS Board of Governors under similar conditions.

## 11.2 Spring maneuvers

On March 1, Amr El-Kadi (the professor in Egypt) wrote Gotterbarn asking for an update “on the response you get from the Steering Committee on V 4.0 of the Code”. El-Kadi wanted to “know how that ends.” He also wanted to know about “the other two task forces: have they completed their work” and “in what form would the steering committee merge the work of the three task forces?” Gotterbarn did not respond until March 16 (apparently using the two weeks intervening to learn what El-Kadi wanted to know). The response, a page long, provides a good summary of the state of the Steering Committee (and its work) at that time. Though addressed to the listserv, it preserved El-Kadi’s email at the bottom and began by acknowledging “the following questions from Amr”. What followed immediately was the wisdom of an experienced committee chair: “Life is not simple.” Gotterbarn then reported that on December 16 he sent out “our recommendation” that Version 4 be forwarded to both “the IEEE-CS and ACM,” that “both societies had meetings of their executive committees in late January and early February”, that he had received no response from the Steering Committee to the December recommendation, and that Version 4 had *not* been forwarded when he checked in January. He had sent a “follow-up letter to the Joint Steering Committee” in January and later “telephoned the ACM co-chair [Dennis Frailey]”. Frailey had told him that the Steering Committee “had not yet formally responded to my (our) request(s).” The Committee had not even responded to Frailey’s requests. More phone calls revealed that the Steering Committee’s “IEEE-CS chair [Cabrera] has resigned”![14] By the time Gotterbarn learned of Cabrera’s resignation, it was the middle of February and both the IEEE-CS Board of Governors and the ACM Executive Committee had met and “we were not on their agendas”. Gotterbarn “spoke with the President of the IEEE-CS in March”; she had “asked someone to be the new IEEE-CS co-chair”.[15] Gotterbarn was still trying to find out when the next Board of Governors meeting would be, but had been put on the agenda of the next ACM Executive Committee meeting in the US—in May, in Washington, DC.

That was the simpler part of Gotterbarn’s administrative life. The email now explained some of the complicating factors. At least “part of the problem we are having” seemed to be “related to the status of the education and skills task force products”. The “skills task force”

(Body of Knowledge) had "spent a lot of money" but had not finished its work. "Some people" wanted to hire a professional group to finish it. Meanwhile, the "education task force" (Curriculum) had been "waiting on the skills task force results, but I think that has currently changed—along with the leadership of the education task force." Gotterbarn then directed anyone interested to a website with reports from all three task forces. (The reports there, except for SEEPP's Version 4, were then all almost a year old.)[16] The change in leadership might not end the Curriculum task force's troubles, Gotterbarn added. The "U.S. based accrediting agency—ABET" had asked the *IEEE* (but not IEEE-CS) to "define a software engineering curriculum". So, it seemed, "there are two distinct efforts going on related to curriculum."[17] The Steering Committee seemed to need more than a new chair.

Before promising to "keep trying to work through the steering committee", Gotterbarn reported that he was "doing my best to publicize the Code." He had had some successes. The Code had been "adopted by some industries" and will soon "appear in some text books". It is "on a web site in Bosnia" and will "appear in some publications". He was hoping it would become "a defacto Standard, [so] that it will be easier for it to get support from the societies."

While Gotterbarn was willing still to work through the Steering Committee, he was no longer willing to work through it alone. He had to be prepared for the Committee again to fail to do anything. So, on April 8, he addressed a formal letter—"Dear Charles House [ACM President] [18] and Dennis Frailey".[19] The letter "respectfully requested[ed] that [Version 4] be considered at the next Executive Committee meeting in May for adoption by the ACM." A similar request was going to the IEEE-CS in time for the June meeting of its Board of Governors.

That was only the beginning of a letter that went on for more than a page. Its second paragraph explained that SEEPP had sent the Steering Committee Version 4 in December and followed up with another letter in January. (SEEPP's letters of January 19 and December 16 were appended in case there was any doubt about what they said.) "We understand," Gotterbarn continued, "the Steering Committee's response has been delayed, owing to reorganization." That delay had "several unfortunate consequences". One was that the "computing community has criticized this joint effort as going nowhere" when in fact there was one, if only one, "finished product", the Code. Second, the Steering Committee's efforts were in danger of trailing events. At a recent conference at which licensing software engineers in Texas was discussed, the Code "was described as a significant contribution to professionalization and licensing of software engineers." Would it not be unfortunate if the Texas Board adopted the Code as their standard before it was approved by the sponsoring societies"? Third, the Code was already receiving "international recognition". That recognition included adoption "by industry such as Siemen's Software Development, India"; the Code's publication in several computer ethics text books ("including Kevin Bowyer's revision of his IEEE-CS book on Computer Ethics"); discussion of the Code at several professional meetings; and its forthcoming publication "in the Journal of Business Ethics, Journal of Science and Engineering Ethics, the Proceedings of the ACM Policy Conference in May (in Washington, DC), and the SIGCSE Bulletin."[20] Would it not be unfortunate, Gotterbarn seemed to ask if, with all this support for the Code, the Steering Committee's inaction were allowed to stop the ACM and IEEE-CS from endorsing it?

The letter then offered three reasons, each having its own paragraph, for the ACM (and IEEE-CS) to endorse the Code. First, the Code had "undergone extensive review by those who will be affected by its adoption" as well as "by ethics officers of large industries, philosophers, and ethicists". Version 4 "represents a general consensus". Second, failure to endorse would

indicate “a lack of support for ethical practices”, forcing “each software practitioner to re-invent the morality of software engineering”. Third, the ACM should “take a stand to provide all software engineers with an ethical direction and with a tool they can use to educate others—including their management—about the ethical responsibilities of the software engineer.” The Code did what the ACM wanted it to do, “clearly state the moral commitments of practicing professionals and of the profession.”

A few hours after sending off this letter, Gotterbarn had a response—from Frailey.[21] He wished that “you had contacted me before sending this [because the] steering committee is preparing a recommendation to the two bodies you mentioned.” That was good news—even if tardy—and more followed. There was “consensus, at least there appears to be, that the code of ethics part [the short version] is suitable for recommending.” Frailey then got to the bad news: “there is some concern that the professional practices part [the main code] is not ready for recommendation, mainly because some steering committee members feel the field is too immature to be recommending such practices.” That news raised many questions. It was Mary Shaw (like Zweben, one of the original ACM-appointed members of the Steering Committee) who typically argued from the premise that software engineering was “too immature” a discipline to be a profession. She had been arguing from that premise for almost a decade. Now, even with support at 95% or above for most provisions of Version 3 (and therefore, probably, for most provisions of Version 4 as well), she had, it seemed, no second thoughts. She was still talking as if there were no consensus on standards of practice. She had, however, (apparently) not remained entirely unmoved. She was now willing to accept the “code of ethics” (although the vote on Version 3 had not included it, that is, the “short version”); what she was still not willing to accept was the “code of practice” (the main body of the code on which a clause-by-clause vote had been taken). What part, if any, did evidence play in her decision? The opinion of practicing software engineers? The “immaturity of the field”? Was she alone in her opposition to the full Version 4? Was Frailey again with her? Did she have other allies? (“*Some* steering committee members” sounded like more than one, but perhaps Frailey wished to give that impression simply to protect the confidentiality of deliberations.)

The Steering Committee was (Frailey continued) “trying to work this out” and planned to “make a formal recommendation to the two councils in May.” Frailey had only been “waiting to contact [Gotterbarn] further until more members of the steering committee had participated in the discussion of the committee’s final recommendations.” What did that mean—“more members”? The Steering Committee was a not large body. It had ten members—not counting the ex officios (who were, apparently, not to be part of these deliberations). Or, rather, it had no more than nine members until Cabrera was replaced. If Frailey was (as he said) still trying to get more members to participate, the consensus he described must be among only a part of the Committee, perhaps a small part, a fragile consensus and one in which even one articulate member (such as Shaw) might carry great weight. There was, then, still no guarantee that the Steering Committee would do anything in time for the May meeting. Frailey was in effect telling Gotterbarn that the Steering Committee was barely operating—and, therefore, that going around it had been wise. Gotterbarn’s letter to the ACM might protect his time on the May agenda; it certainly had made it harder for the Steering Committee to do nothing or to send just the short version of the Code forward. Everyone at the ACM Executive Council would know that SEEPP had recommended the entire Version 4, not just the short version, and had done that in December (almost five months before the May meeting).[22] Gotterbarn had, it seemed, out-maneuvered the

Steering Committee—or at least a minority of it. There might be awkward questions at the Executive Council meeting, but at least the complete Version 4 would be in front of them in May. He could make his apologies then.

Having offered Gotterbarn some hope, Frailey returned to a subject about which Gotterbarn had worried since September 1997, the dissolution of the Steering Committee. Frailey posed (what he called) "one simple question": "Should the code of ethics activity continue and, if so, what bodies of ACM and the computer society should sponsor it?" The Steering Committee was going to recommend that each of its sub-activities "be continued, if appropriate, to be jointly sponsored by the appropriate bodies in the two societies, with a new 'coordination' body to be formed as a replacement for the steering committee". Of course, Frailey's question was not simple. The coordinating body could be a permanent version of the Steering Committee under Barbacci, active and helpful, or a presence even more shadowy than it had become under Cabrera, or any number of other things—better or worse than the Steering Committee had been. Everything depended on the details: membership, budget, powers, reporting line, and chair. Though Frailey offered no details (and seemed to have none), Gotterbarn could see that the Steering Committee's dissolution might not be the disaster he had anticipated. He would have to think hard before he answered Frailey's "simple question".

A few days later, Gotterbarn learned that Leonard Tripp had replaced Cabrera as Chair of the Joint Steering Committee. Though Tripp had been a member of the Steering Committee since early 1996, Gotterbarn had never met him. (The Steering Committee had held only one face-to-face meeting during Cabrera's term (November 1996), holding that one without ex officios—except for Melford.) Gotterbarn knew Tripp only from one phone conversation back in 1994. Melford had called Tripp, one of the "IEEE-Standards gurus", during the April 24 meeting in DC, to ask about how to write a "PAR" (3.4). A distant voice coming through a speaker phone set in the middle of the conference table, Tripp had seemed a true "organization man".[23]

Had Gotterbarn learned more of Tripp during the intervening years, he might have found much to confirm that first impression. Though a member of the ACM as well as of the IEEE-CS, Tripp had largely confined his professional activities to the IEEE. He had served on the Operations Committee of the IEEE-CS Technical Council on Software Engineering, chaired the IEEE-CS Standards Activities Board, Standards Coordinating Committee, and Software Engineering Standards Committee. He also chaired the US Technical Advisory Group on software engineering standards. Had Gotterbarn known all this, he would have had no trouble understanding why Melford considered Tripp "Mr. Standards". He might, however, have wondered why Tripp took such an interest in Standards. The explanation could not be Tripp's education. Both his degrees were in mathematics (Brigham Young University, BS and MS). As a mathematician, Tripp *should* have gravitated toward ACM activities (and away from Standards) rather than toward IEEE-CS (and Standards). What seems to explain his preference for the IEEE-CS—and for standards—is his employer. Tripp had worked for the Boeing Company for thirty years. Boeing is an engineering organization *par excellence*, building some of the world's largest, most sophisticated aircraft, both civilian and military. Technical standards, including standards for software, are central at almost every step of what Boeing does. Tripp had (like many other software engineers) moved from thinking of software as mathematical formulas to thinking of it as the most complex of machinery, ethereal but vital. His specialty had become standards for safety-critical airborne software. For him, the Standards-setting process was a matter of life or death.[24]

Gotterbarn worried about having "Mr. Standards" in charge of the Steering Committee. For him, the term "IEEE Standards" meant meetings concerned with PARs, CFPs, the Operations Guide, Scopes, and other procedural matters that bored him, produced nothing but procedural documents, and seemed to draw nothing but yawns from ordinary volunteers. Gotterbarn recalled Chikofsky's 1994 warning that the IEEE Standards process could go on for "a long time" (3.4). Since that warning, Gotterbarn had learned that "a long time" meant "years"; he had himself slaved three years in the salt mines of that complex process.[25] When Melford resigned in December 1996, Gotterbarn felt as if chains had been struck from his legs. He had abandoned the IEEE Standards process, replacing it with something like the simple process by which the ACM had adopted its code. He had accomplished more, infinitely more substantively, in one year in that way than he had in the three preceding years following the IEEE standards process. Was all that had been achieved in 1997 to disappear into the endless tunnels of that dark process?

Since Tripp was now in charge, there was nothing to do but to try to work with him. So, a few days after learning of the Steering Committee's new chair, Gotterbarn wrote Tripp "a status report" clearly designed to show him that the IEEE process had, in effect if not in form, already been satisfied. "We" worked on the code for "almost two years" (counting, it seems, Mechler's efforts in 1996 as well as his own in 1997, but not 1994's procedural activities). There had been "a broad based development effort—truly international in participation—with contributions from a wide range of practitioners." Having framed the process in this way, Gotterbarn took a page to summarize what was already in the December and January letters of transmittal to the Steering Committee and in the March letter to the ACM (the reasons to move quickly). That letter is dated April 20.[26]

Early Sunday morning, April 26, Gotterbarn sent Mechler the standard email informing him that (in response to his report of a new email address) he had (once again) been added to the PRFCMP-L mailing list. A few minutes later Gotterbarn added a personal note (apparently, to answer Mechler's "what's new?"). "[Trying] to manage a list at another site" is, Gotterbarn observed, "no fun", but Mechler had not missed any messages. There was news, though. There had been "some leadership changes on the steering committee". Gotterbarn was "working to get the Code version 4 in front of both the IEEE-CS and ACM executive committee meetings." Then, changing the subject, Gotterbarn asked whether Mechler knew "of anyone using the Code." Gotterbarn knew "of one Seimen's shop, an insurance company, a division of Hitachi, one army training program and that is about all I remember at this moment."[27] Mechler seems not to have responded.

Two days later (April 28), Gotterbarn sent an official letter ("Dear President House") requesting the ACM Executive Council to consider adopting Version 4 at its May meeting. The letter exists in two electronic forms. One is ordinary text; but the other is on the letterhead of Gotterbarn's Software Engineering Ethics Research Institute.[28] Using software just becoming available, Gotterbarn had made his electronic mail as impressive as possible. He sent the letter and a copy of Version 4 (as he said in a covering memo) as "support material for the ACM's meeting in May". The letter itself indicated he did this on Frailey's instructions.[29] Though much the same as his earlier letters of transmittal, this one included one new item with the title "Software Alliance". The Texas Board of Professional Engineering Licensing Committee was, he reported, "considering the Code for their proposed licensing standard for Software Engineers". Following that startling news, he listed the Code's recent successes: It was being

used in a program for industrialists in Australia, at Army facilities at Fort Monmouth (New Jersey) "in training materials for computer engineers, computer scientists, and computer specialists", and so on.

On May 5, Gotterbarn heard from IEEE-CS. After several days of "telephone tag", he and its new president, Doris Carver (co-chair of the Curriculum task force, 1994-96), had finally been able to talk. She later sent an email confirming the substance of the conversation (with a copy to Tripp).[30] The "procedure you will need to follow is for the report to be submitted via the steering committee". The IEEE-CS half of Gotterbarn's attempt to bypass the Steering Committee had failed. Carver pointed out (what she—correctly—thought Gotterbarn knew already) that Tripp was the new Steering Committee chair. "[The] best path is", she advised, "for the two of you to communicate." Gotterbarn decided to wait until he knew what the ACM Executive Council would do on May 10. If they approved the Code, he would be in a better position to work out something with Tripp.

By the time Gotterbarn attended the ACM's Executive Council Meeting on May 10, he knew that he had underestimated the Steering Committee (or, at least, Tripp). No longer a "silent partner", it too had put an item on the Council's agenda.[31] Though titled "Report of Task Force on Software Engineering—Software Engineering Code of Ethics & Professional Practice", the item was in fact a status report on the work of the Joint Steering Committee. John Werth (Carver's ACM counterpart on the Curriculum task force, 1994-96) spoke for Frailey (who was absent). After listing the four purposes for which the Steering Committee had been established in 1993, Werth reported that the Engineering Licensing Committee of the Texas Board of Professional Engineers had approached the Steering Committee in 1997 for assistance with "the issue of software engineering licensing".[32] The Licensing Committee had "formally requested that ACM and IEEE-CS jointly work with it to define a body of knowledge on which national engineering licensing exams [for software engineering] will be based." The Texas Board of Professional Engineers would next meet in Fort Worth, June 17-18, and might then enact rules for licensing software engineers in Texas. The question for the Council was what to tell the Texas Board at their June meeting. The Steering Committee knew at least as much about events in Texas as Gotterbarn did.

The minutes report one Council member or another observing that "coming out as a discipline will be a challenge" for software engineering, that there "will be a need for strong industry representation", that it was "not known whether industry was driving this issue", that the Texas Board had already set criteria for licensing disciplines "less well defined than SE", and that software engineering had in recent years changed from a "supporting activity" to a "princip[al] player." When the discussion ended, the Council unanimously took two actions. First, it declared that the "ACM would be pleased to assist the Board in defining the body of knowledge and standards of practice on which such licensing exams would be based [should the Texas Board vote to adopt rules for licensing software engineers]." The ACM would also be "pleased to work with the IEEE CS in this endeavor." Second, the Executive Council requested that "Dennis Frailey or Leonard Tripp, in consultation with other members of the Steering Committee and, if possible, including participation from SIGSOFT and the ACM Education Board, propose a management structure whose responsibility would include: 1) coordinating follow-on activities of the Steering Committee in the body of knowledge, ethics, professional practices and education areas; 2) assisting State Licensing agencies…; 3) assessing the implications of licensing on current software engineering practitioners and ameliorating any

negative implications." The proposal was to be brought back to the ACM Executive Council at its next meeting (June). It was now official. Something more permanent was soon to replace the Steering Committee (and it would have the power to work with state agencies much as Buckley had suggested early in 1993). But what that something would be remained to be decided—and it would be decided not by the Steering Committee as a whole but by Tripp and Frailey (after "consulting" with the Committee and others). Apparently, Tripp and Frailey were to have a free hand; the Steering Committee was already (more or less) dead.[33]

The ACM Executive Council, fifteen members present, met on Sunday at Washington's Renaissance Hotel from 8:30 AM to 4:10 PM (May 10, 1998). The part of the meeting devoted to Werth's report, the subsequent discussion, and the two motions was (judging by the minutes) about one thirteenth of the whole (30-40 minutes). A slightly larger part concerned Gotterbarn's report. The time set aside for the Steering Committee at this meeting (that is, for Frailey's report as well as Gotterbarn's) was more than a quarter of the whole, evidence of the importance the ACM assigned the Committee's work at this time.[34]

Gotterbarn began his presentation by explaining that the Code was a set of standards for a "sub-specialization" within the ACM and IEEE-CS, that the proposed draft (in one version or another) had been published on the web, in IEEE-CS's *Computer,* and the ACM's *Communications,* with a ballot to be returned, and that most provisions of the code had received a 95% approval in the balloting that followed. Gotterbarn asked the Executive Council to approve the Code more or less as it had approved the ACM Code six years before.

The discussion that followed did not go as Gotterbarn had expected. He soon learned that the Steering Committee had made its own recommendations concerning Version 4: Version 4 was "ready for review in a formal process". The process "should include balanced selection of representatives from software practitioners, educators, producers, and users" and "should follow the principles of due process such as those used in standards development."[35] Upon completion of the formal review, the Code would be "suitable for consideration by the appropriate bodies". The "appropriate committees of ACM and IEEE CS" should conduct the formal review. It was this motion, as Gotterbarn learned after some confusion, not his, that the ACM Executive Council officially had before it.[36] Gotterbarn also learned that Tripp was now IEEE-CS's President-Elect (that is, the President for 1999). "Mr. Standards" was now formidable indeed.

The rest of the discussion, though surprisingly peripheral, probably made the Executive Council hesitate to do anything definitive. One Council member, noting that the Code did not define "software engineer", asserted (perhaps in answer to Gotterbarn's description of software engineering as a "sub-specialty") that in Europe software engineering was "an integral part of Computer Science". Gotterbarn did not deny that, but reported that in Europe, or at least in Britain, computer science was itself considered part of engineering. That answer may have suggested the next question: If the ACM Code of Ethics is for computing, why have a separate code for software engineering? How is software engineering different? (The implicit premise of the question seems to be that a "sub-specialty" may, or may not, be different enough from the broader discipline to need its own code of ethics.) Gotterbarn responded that the Code "contains practical details of what makes SE an applied science". The response seemed to satisfy the Council on that question; the questions that followed were about the consistency between the Software Engineering Code and the ACM and IEEE codes. Had anyone checked for consistency? Gotterbarn could not say that anyone had but tried to dispose of any worry by pointing out that the Code's "purpose is to help the profession and not only ACM's membership

[the purpose of the ACM code]". The two codes did not apply to the same people. There was nonetheless agreement that a check for consistency should be done before the Executive Council approved the Code. Eventually, the Council unanimously asked COPE (the Committee on Professional Ethics) to "review the proposed [code] for compatibility with the ACM Code… and to determine the appropriate further process, if any, prior to adoption by the Council." At that time, COPE had three members. One was Robert Riser, one of Gotterbarn's colleagues at ETSU; a second was Keith Miller (a member of SEEPP's Executive Council); and the third, the chair, was Gotterbarn himself.[37] The Council had postponed approval but had refused to decide between Tripp and Gotterbarn. Like the IEEE-CS President, it had, in effect, told Gotterbarn to work things out with Tripp. That was not the worst outcome imaginable but—as Gotterbarn later said—"It was not a happy moment for me."[38]

## 11.3 Super-fast Standards process?

Gotterbarn now had no choice but to talk to Tripp. This he did soon after returning to Tennessee (probably early May 14).[39] The phone conversation began with Gotterbarn having no idea what to expect. He did not even know whether Tripp favored or opposed the Code. Yet, Gotterbarn's first impression of him was favorable. Though soft-spoken, Tripp clearly knew what he was doing and was business-like about doing it. He saw his job as supporting the production of the best—"most effective"—code possible. He had the same commitment to the Steering Committee's other task forces, Curriculum and Body of Knowledge. He wanted to make software engineering a profession. Following the IEEE Standards process was part of doing just that. The Code would not have much chance within IEEE-CS if anything but that process were followed. Engineers do not like to adopt Standards ad hoc. Tripp did not, however, expect the adoption process to take years. He had overseen the adoption of many Standards. He saw no reason why the whole process for Version 4, including approval by the IEEE-CS Board of Governors, could not be complete by autumn. The crucial step was getting the ballots out. Once they were out, the process would have a schedule—and automatically produce "closure" by a fixed date. Of course, to get the ballots out, "we" (Gotterbarn and Tripp) would have to develop a balanced list of reviewers, people whom both IEEE-CS and ACM members would respect as individuals and who, as a group, constituted a fair representation of the relevant constituencies. But Tripp had a staff experienced in developing balanced review groups, lists of potential reviewers, and so on. If Gotterbarn would prepare a preliminary list, including everyone he thought might be appropriate, Tripp would add whatever was missing. Tripp wanted the invitations to participate in the review group out by the end of May.[40]

By the time Gotterbarn hung up, he felt relieved and tired. He felt relieved because Tripp had seemed someone he could work with—competent, calm, with his heart in the right place—and because Tripp had an experienced staff to do much of the work. The nearest Gotterbarn had come to that sort of support was a student or two. Gotterbarn nonetheless felt tired after the conversation because he had not planned to devote his summer to getting the Code through the IEEE Standards process. He had promised to write several papers, had an NSF grant proposal to prepare, and was teaching summer school. He felt tired too because he doubted that anything as complex as the process Tripp described could be carried through as quickly as Tripp said it could.[41] Everything in Gotterbarn's experience told him to double or triple the time estimates on a project like that. But Gotterbarn had no alternative. If the Code were ever to be adopted, he had

to do as Tripp said. Gotterbarn quickly phoned Carver (IEEE-CS President) that he was working with Tripp.

A few minutes later, Tripp emailed Gotterbarn a one-page draft of "guidelines" and a draft ballot (with "15 July 1998" as the deadline for the end of balloting).[42] Later that day, Tripp also sent a draft "Invitation to Ballot" (with "1 June 1998" as the date to send it out). The arrival of these documents seemed a good omen. The next morning Gotterbarn wrote "Leonard",

> Boy do I like your efficiency!" The process you outlined looks fine. I have attached the three files you sent to me with some suggestions for modification. Let me know what you think.
>
> I am still developing a list of ballot invitees. I should have it in you[r] hands by Monday [May 18].[43]

Beneath Gotterbarn's signature (the usual "Don") were "Comments on the three documents". He had used "the comment function of word [MSWord]" to insert specific comments (another innovation in software). Anything he wanted to delete was in brackets and additions in quotes. He thought it important to use the version number of the Code since some of those voting on this version may have voted on Version 3. All references to the Code should be capitalized. He had some questions about the list he was preparing: "Are we still going with the categories of interest being: User of software engineered product[;] Producer of software engineered product[;] General (both user and producer)?" He wondered how to identify "Professional Engineers" (that is, licensed engineers). He even wondered what the official name of the Steering Committee is. (At the end of the Invitation, Tripp had referred to the committee he chaired as the "Software Engineering as a Discipline Steering Committee" after getting the name right at the beginning.)[44]

Within a few days, Gotterbarn sent Tripp about 270 names (with email addresses) drawn from (as he said in a note to himself) "various computer societies, cross section [of] people involved with licensing, Consultants, Academics, Professional Engineers, Consumers 'shrink wrapped', Custom software [users], anyone who responded to code, people from insurance company, Hitachi, Croatia, peter neuman."[45]

A few days after that, on May 26 (four days ahead of schedule), Tripp emailed the official invitation to participate in the balloting. It came from "Leonard L. Tripp, Chair, Joint Steering Committee for the Establishment of Software Engineering as a Profession", went to more than two hundred recipients addressed as "Dear Colleague", and had as the sender's email address twoods@computer.org (evidence, to those who knew about such things, that Tripp had staff support at IEEE-CS's Washington headquarters). The letter began with an invitation from the "IEEE Computer Society and ACM" to participate in the "formal review of version 4 of the Software Engineering Code of Ethics and Professional Practice." (The mention of ACM, and the inclusion of prominent ACM names on the list, was, it seems, intended to make a separate ACM ballot of membership unnecessary.) The letter briefly explained how the Code fit into the larger effort, begun in "November of 1993", to establish software engineering as a profession, concluding with the significant statement that the SEEPP task force "has completed its work and recommends that the Code be reviewed by a panel of peers using a formal review process."[46] The Code (Tripp continued) "addresses both the responsibilities of the practicing professional and of the profession". "We" (Tripp, SEEPP, the Steering Committee, or perhaps all) believe that

the societies "representing the profession" should "provide all software engineers with an ethical direction and with a tool they can use to educate others—including their management—about the ethical responsibilities of the software engineer."[47]

Having set a high tone, Tripp explained what participating in the "balloting group" entailed. The balloting group "will have 20 days to review and respond to the draft code." Reviewing the Code would be demanding: "The draft code is 7 pages long." The twenty days would probably be between June 22 and July 15, 1998. If interested, "you" should fill out the form at http://ada.computer.org/Ballot/Index.htm by June 12. By filling out that form, "you certify that you understand the subject matter of the proposed document and are technically competent to cast a ballot" Filling out the form also means "you agree to respond in the specified period". Responding in time would be important. The ballot would not be valid unless "at least 60% of the ballots sent [out]" were returned in time. For that reason, "members of a balloting group had an obligation to respond." The next to last paragraph provided further administrative detail (and directed all questions to "Tracy Woods, Volunteer Services Coordinator, IEEE-CS"); the last paragraph simply thanked the reader for "your time and participation".

Mechler received this invitation just before 6 PM on May 26. The next day, about noon, he received a short but hopeful follow-up message from Gotterbarn (sent through the listserv):

> You have all probably received an "Invitation to Ballot". This is the last formal step in processing the code. I hope you will all volunteer to be part of the balloting group and vote. The entire process—setting up the balloting group—sending [out] ballots, responding to negative ballots, producing a formal recommendation from the steering committee—is scheduled for completion by September—1998!

Around the world, others involved in software engineering were receiving Tripp's message—and some were also receiving Gotterbarn's. But at IIT, only Burnstein received Tripp's message—and, apparently, no one received Gotterbarn's. At least, we have no record of Gotterbarn's follow-up—or, indeed, of any other messages sent through the listserv in 1998.[48] This is odd because Gotterbarn and I were still corresponding all through 1998. Indeed, the day after he sent his follow-up message (May 28), I thanked him for "written comments" on an article he was "writing" for CSEP's newsletter *Perspectives*. Gotterbarn did not actually have time to write what I wanted. So, I interviewed him by phone, wrote up the interview as an essay of about 1500 words, and sent it to him for corrections, additions, and approval. (We had discussed the draft by phone, but he had also sent written comments.)[49] My note continued: "Some of the information [in the article] may also be useful later—if I ever get around to doing a book on professional codes, with your experience as the central example—an enormous success if the ACM and IEEE ever approve the code." I then promised to keep him informed concerning plans for what became this book "not least because I would want to do some interviews to fill in the many holes in the process as I know it—and to provide a useful step-by-step handbook of how to do it [write a code of ethics]." I cannot tell whether this letter was faxed or carried to its destination on a human shoulder. What seems probable is that I did not use email. Though the header includes an email address for Gotterbarn, it is an address that, by May 28, 1998, was almost a year out of date (Gotterbarn@ACCESS.ETSU.EDU). Communication between SEEPP and IIT (though not between Gotterbarn and me) had broken down—again.

## 11.4 The Vote

The breakdown of communication between SEEPP and IIT is one indication that selecting the balloting group may not have gone quite as planned. There were others. On May 27, Laurie Werth responded to Gotterbarn's email: "I leave town on June 6 and return on Aug 5—I fear that this will disqualify me unless I could vote before I leave. Is this possible?" We have this email because it survived in Mechler's files.[50] It did not survive in Gotterbarn's even through Werth sent it through the listserv. There is an explanation. In response to the writing of this book, Gotterbarn asked the ETSU system for a copy of his emails. What it did instead was to copy only the headers for 1998, deleting as it did all the content of the corresponding emails.[51]

Mechler apparently thought that the pool of potential voters should be as large as possible. He therefore forwarded the letter Tripp had sent him to anyone he thought should vote. Melcher's cover memo would say (something like):

> Received the invitation to ballot below and thought you may be interested. Yesterday I was on the web and applied. It looks like any one who visits the site or goes to it through IEEE CS site can ballot. Here are the web sites for the committee and the code if you want to look ahead of time.[52]

Was Mechler alone in sharing his invitation? How many voters entered the lists in this way? We do not know. What we do know is that such volunteers did not invalidate the process. The crucial step in achieving a balanced review panel was the final assembly of the group. The IEEE had standards for assuring an appropriate balance. If too many volunteered from any one group, only some of the volunteers would be given a ballot. If too few volunteered from any group, others from that group would be sought.[53]

When Mechler next wrote Gotterbarn, it was not to brag about how many potential voters he had signed up but to report that he "had [on June 29] tried to set confirmation on the list and got this response ['you are not subscribed to the PRFCMP-L list']." Observing that he seemed "not to be set", he asked whether he had missed anything. A few hours later, Gotterbarn informed Mechler that he had indeed missed something. The "[balloting] process is underway." The voting role had 192 names on it, of which Mechler's was one. "You will be getting email shortly about the web site we are setting up for the balloting." Gotterbarn then got around to the problem that had provoked Mechler's email. He had just "checked the prfcmp-l list and your name is listed as mechler_edmund@ADTRANZNA.COM." So, Gotterbarn continued, the listserv would not recognize "you" since "you" were using a different email address.[54] The next morning, Mechler thanked Gotterbarn "for the info" and explained how he came to ask the technical question Gotterbarn had answered the day before: "I never look at my e-mail [address] because everywhere else I have been, changed jobs three months ago, I had an internal and external e-mail address." But, where he was working now, he "found that I have another one for an unknown reason and during a change a couple of weeks ago the other was used when I sen[t]." He was just then "getting it fixed". He apologized for any trouble he had caused Gotterbarn, sketched what "set confirmation" meant (in response to a question from Gotterbarn), expressed satisfaction that the balloting was beginning, and concluded with details of how his IEEE-CS email address forwarded mail to the address he was actually using.[55]

The next morning (July 1), Mechler received Tripp's second "Dear Colleague" letter. This one informed him that he had been "selected to participate in the formal review" of the Code ("based on your response to our original Invitation to Ballot email in May"). Then, after taking a paragraph to repeat the history of the Steering Committee, Tripp instructed Mechler to go to a certain IEEE-CS web site where he would find three items: 1. Ballot Guidelines; 2. the Code; and 3. the Ballot Form. Tripp suggested taking "a moment to carefully read the Guidelines before proceeding to read the actual [Code]", referred all questions to "Tracy Woods at IEEE-CS", and stated that the deadline for return of the ballot was "Monday, 20 July 1998". The last two paragraphs were identical to those in his letter of invitation.[56] Tripp did not say it, but those who had kept his first letter could see that balloting was not quite on schedule. Tripp had said that it would (probably) take place between "June 22 and July 15, 1998". It was now scheduled to end five days beyond the bracketed dates.

On July 10, Woods sent out a "reminder" that the "deadline to respond to the Code of Ethics is Monday, 20 July", stressed the importance of getting a response rate of "at least 60%", and then pointedly reported that (half way through the twenty days of voting) "we have only reached 22%". She urged those with a ballot to "respond even if your vote is an abstention". Below her note she had helpfully placed Tripp's July 1 letter giving instructions on how to vote.

Early in the afternoon of July 20, the last day of voting, Mechler emailed Woods, "Did we make it???" Just after 8 AM the next morning, she replied, "Yes, as of Monday morning we had a 66% response rate. More came in throughout the day and overnight." The final response was 77% (148 out of 192 possible).[57] The vote was valid.

The votes now had to be counted, comments extracted, and a response devised for each comment. A computer at IEEE-CS automatically took the ballots off the web page, counted them, and placed each (including all comments) on a six-column spreadsheet (the first column for the number of the ballot, the second for date and time received, the third for the voter's name, the fourth for the vote, the fifth for "comments", and the sixth for "additional comments"). Version 4 had done well: 77% (113 of the 148) voted Affirmative, with 45 of those offering an optional comment. Only one percent (2) had voted Abstain. The rest 23% (33) were Negative.

Had Version 4 done as well as Version 3? Clearly, if we compare only raw percentages, the answer must be no. Support had dropped from 95% to 77%. But there are at least three reasons to discount such a comparison. First, those voting on Version 4 were not the same as those voting on Version 3. They may not even have been the same sort of people. Those voting on Version 3 consisted largely of IEEE-CS and ACM members who received the relevant society's general publication, read an article on ethics, and (with no other encouragement) filled out an eighty question survey. They paid $100 year to be a member of one (or both) of those societies. Though Gotterbarn had attempted to bring in some groups (primarily, corporate officers) who might not fit this (uncommon) profile, he had not systematically aimed at "balance". Tripp had. The overlap between those voting on Version 3 and those voting on Version 4 could not be more than 27%—and might be as little as 7%.[58] Those voting now may well have included many who, though willing to vote yes or no, would not have taken the time to fill out the elaborate survey the vote on Version 3 required. Those voting now may also have included many who voted out of devotion to the IEEE's Standards process, that is, people who voted because they received a specific emailed invitation. It certainly included two who, though they could not properly have voted on Version 3, happily voted on Version 4, Gotterbarn and Miller.

That is one reason to think the 77% favorable vote on Version 4 does not represent a loss of support when compared to the 95% vote on Version 3. Another is that there was no overall vote on Version 3; only on individual clauses. For all we know, many people who voted against even one clause would, if asked about the Code as a whole, have voted no. There is nothing irrational, or even unusual, about one clause in a complex document being a "deal breaker". Equally possible is that many who voted against many clauses might nonetheless, if asked, have voted for Version 3 as a whole—on the principle that, whatever its defects, Version 3 was much better than nothing.

A third reason not to treat the vote on Version 4 as comparable to that on Version 3 is that the voting procedure was different. Those voting on Version 4 knew (or, at least, should have known) that their vote and comments would be public (that is, posted on the IEEE-CS web site for all to see).[59] Those voting against the Code had to explain their vote (in the comments section)—or the negative vote would count as an abstention. They could not just complain; they had to propose an alternative (if only deletion of the offending language). They knew (or, at least, should have known) that if SEEPP accepted their alternative, their negative vote would automatically become positive. They would get another opportunity to vote only if SEEPP made some change in response to the negative comment but not exactly the change suggested—or only offered reasons not to make any change. Voting negative on Version 4 was part of a complex public conversation aiming at consensus, not a mere survey response (as the vote on Version 3 had been). Voters might well vote differently in environments so different.

While those three reasons seem to make a strong presumptive case for not treating the vote on Version 3 as comparable to the vote on Version 4, there is some reason to think that the two votes might nonetheless be comparable. In a few cases we know how a voter would have voted on both versions. Six people—Burnstein, Little, Langford, LaRue, Maner, and Mechler—voted in favor both times (or, in the case of Version 3, seem to endorse the code in such a way as to tell us that, if they had been asked, they would have explicitly endorsed the whole); and, of course, two more—Gotterbarn and Miller—would have done so had they thought to vote on Version 3. More interesting are the four—Berleur, Black, Harding, and Sigut—we know to have voted no on Version 4 and to have expressed a strongly negative opinion concerning Version 3.[60] All four wrote extensive comments as part of the balloting on Version 4; all complained of earlier suggestions not taken; and none complained about anything added between Version 3 and 4. They seem to have voted against Version 4 for the same reasons they would have voted against Version 3. Most interesting of all is the remaining category: those known to have voted yes on one version and no on the other. That category seems to be empty. We are then entitled to conclude that voting on the two versions was comparable enough to preserve consistency in response. What might comparing the two results tell us?

It will not tell us that Version 4 had lost 20% of the support Version 3 had. We cannot know about that without doing a comparison of all the people in the two voting pools—along whatever are the right axes for predicting how they would vote. I do not know how to do that. What comparison of the two votes does tell us, if it tells us anything, is that all the work that turned Version 3 into Version 4 probably had little effect (positive or negative) on the support Version 4 received. The category of yes-on-one-and-no-on-the-other is empty. Had Version 3 been put to the same test as Version 4, Version 3 must (assuming the category of vote changers really is empty) have received a favorable rating of about 77%. Is the same true of Version 2.1—and even of Version 1.0? That is certainly possible on the evidence we have. Are we to conclude,

then, that had Gotterbarn teamed up with Tripp in November 1996 rather than in May 1998, the whole process could have ended eighteen months earlier?

Mechler once told me that he thought Gotterbarn's various versions of the Code amounted to "moving deck chairs around on the *Queen Mary*". [61] He did not mean that the succession of versions was as futile as the proverbial "moving deck chairs around on the [iceberg-gashed] *Titanic"*. On a ship likely to reach port in a few days, the location of deck chairs matters—but, in large part at least, only to the passengers. Their location will not speed the ship on its way or delay its arrival by even a minute. In writing codes, as in arranging chairs on deck, there is considerable "free play"—affecting the look, feel, and content of the code, but not its likelihood of ultimate approval. At least one other member of SEEPP/E holds a similar view. According to Jayaram, "He [Mechler] kept us going until we drafted the code (Version 1). Everything after that was minor in comparison."[62]

Questions about what might have been are, of course, notoriously hard to answer, especially in human affairs, and I do not claim to know what would have happened had Tripp and Gotterbarn teamed up eighteen months earlier. I do, however, think it worth noting a little more evidence that Gotterbarn's successive revisions may not (overall) have improved the Code's chance of adoption (however much they may have improved the Code). A number of the negative votes cite as one reason (or *the* reason) for the negative some way in which Version 4 differs from one or more of its predecessors. So, for example, George Samaras (#25) objects to "economic disadvantage" in Clause 1.07. "If the software is too expensive," he reasoned, "it will not be used/will not sell." The issue is economic, not ethical. 1.07 made its first appearance in Version 2.1 (as the ever-controversial "diversity" clause 2.05). Or, consider the next negative vote on Version 4: William S. Junk (#27) voted no in part because "1.03 ['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'] does not seem to recognize that there are legitimate applications of software technology that may have negative consequences on the quality of life of some individuals or result in some harm to the environment." Clause 1.03 is the successor of Version 1's 2.02 ("Approve software only if they have a well-documented belief that it is safe, meets specifications, and has passed all appropriate tests"). The chief difference between the two (or, at least, the difference relevant here) is the addition of consideration of "quality of life" and "harm to the environment", the very phrases that concern Junk. Junk argues that there are legitimate projects for software engineers, for example, a "smart weapons system", that cannot avoid reducing someone's quality of life (the target's) or harming the environment (as collateral damage). Junk is nonetheless willing to change his vote if (along with changes in some other provisions), the word "substantially" were inserted before "diminish" and "harm".[63]

We are, however, not entitled to conclude from Junk's objection to 1.03 that the executive committee made a mistake by expanding the coverage of old 2.02 or that it could have increased the total vote for Version 4 by anticipating Junk's amendment. It is at least possible that some positive votes depend on 1.03 not being weakened in the way Junk suggested. Having pointed out that possibility, we can say no more. While the executive committee undoubtedly had a certain amount of freedom in what it did with 1.03, it did not know—and could not have known—how much freedom it had. Indeed, even now we do not know that.

11.5 Can this really be the end?

Balloting closed on July 20. On July 22, Woods sent Gotterbarn the votes and comments. Gotterbarn suggested removing some personal information from two entries. On July 24, when Woods posted the official (slightly edited) spreadsheet on the IEEE-CS website, Gotterbarn was already at work on his responses. He had to respond by accepting each proposed change, by making some lesser change, or by explaining why he (or, rather SEEPP) had left the provision as it was. Gotterbarn had about three weeks according to Tripp's schedule—and only a few days more than that till the start of the Fall Term.

Gotterbarn had at least two reasons not to revise the Code again. First, ninety-three voters had put something in the comment section (with twenty of these also making "additional comments"). Once these comments were organized into distinct points (with one or more people making the same point), each required a distinct response—as well, perhaps, as a revision in the Code.[64] Responding would be a big job, responding in a few weeks, bigger yet; and carefully reconsidering every provision someone challenged, a very big job indeed.

The second reason Gotterbarn had not had to revise the Code again was that Version 4 had passed the formal review process already. According to Tripp, 60% was the minimum to pass.[65] All Gotterbarn could hope to do now was raise the positive percentage above the already comfortable 77%. There was *no* guarantee that any further revision would in fact do that. Votes gained might be balanced or over-balanced by votes lost. Yet, working with his executive committee in a way by now familiar, Gotterbarn did much more than the minimum the procedures required.[66] By August 6, when he met with Miller and Rogerson at the "Tangled Web" conference at Dartmouth College, he had a document sufficiently different from Version 4 to be called "Version 5.0" and a draft of 185 "responses", explaining what had (and had not) been done in response to various comments.[67] The three found time during the conference to review and revise both documents.[68] Then it was off to Boston to deliver a paper on August 10 at the World Congress of Philosophy. Gotterbarn sent Version 5.0 and the comments to Tripp (as attachments) as soon as he returned to Johnson City. Tripp acknowledged receipt about 4 PM, August 12 (Wednesday), asking Gotterbarn to "resend the second file" (the comments) because he could read only the first file (with "version 5" in it). Guessing that the problem was the second file's name, Tripp recommended that Gotterbarn shorten the file name to eight characters or less before trying to resend.[69] Tripp concluded this email with a reminder: "Next step is recirculate the revised document and the comment resolution to the balloting group. The target is to submit the document and the comment resolution to the leadership of the two societies for acceptance."

On August 31, the 185 responses were published in the sixth column of the spreadsheet reporting results of balloting on Version 4.[70] On September 9 (fifty days after the close of voting), Tripp emailed everyone who had participated in the balloting his third "Dear Colleague" letter.[71] It not only reported the results of the first round of balloting but the results of work done since. Thirteen negative votes had been "resolved" (with twenty still "unresolved"), raising the affirmative from 77.4% to 86.3%.[72] It was now time for a "recirculation ballot". The purpose of this ballot was to "confirm that comments during the initial ballot have been dealt with in an acceptable manner" and also to "increase consensus on the Code of Ethics document".

Tripp then directed voters to the appropriate IEEE-CS website, asking them "to review the Code of Ethics (Version 5.0) and to complete the Recirculation Ballot."[73] Those who did not

wish to change their vote need do nothing. Negatives would remain negative; positives, positive. (Silence did not constitute consent, except for those who had already consented.) There were only four reasons to change a vote: First, those who thought all their objections had been "addressed" could change their negative vote to affirmative. Second, those who thought "one or more" of their objections had not been addressed could "reiterate [their] negative vote". Third, those who wished to "indicate [their] support of objections of another person which have not been resolved" could change an affirmative vote to a negative. Fourth, those who thought "changes made to the document have introduced substantive problems that did not previously exist" could change their affirmative vote to negative. The deadline for responding was September 16 (one week from the date this email was sent out). Tripp ended the letter inviting "comments on the balloting process", asking that "the questionnaire on the same Web page" be completed, and directing any "technical or procedural questions" to his own (Seattle) phone number. [74]

On September 23, Woods issued the final tally. She listed thirty-two voters as originally negative.[75] Of these, only twelve had voted again. Half of these had changed their vote to affirmative; half (including Junk and Sigut) had voted negative again.[76] Did any affirmative votes change to negative? The answer *seems* to be no, but even Gotterbarn does not *know*. What he reported to the listserv on October 13, 1998, is all he recalls today: "I do not have the official final word, but 6 of the unresolved negative votes changed to positive so I think the final analysis is 132/146 affirmative (90%). That is a passing grade :-)" He then added that the Steering Committee would be "presenting the Code to the ACM Council….this month and to the IEEE next month….It may be a great Thanksgiving."[77] Seven days later (October 20), he wrote the listserv again: "Yesterday the ACM Council met in Vancouver and approved the Software Engineering Code of Ethics and Professional Practice (5.1) as their approved Code for the practice and teaching of Software Engineering!!!!!!!!!!" The vote was unanimous. There had not even been one abstention. He then added that the "FINAL version" was "on the seeri web site….I am exhausted…….more later."[78]

This email started another listserv party (the first having been almost a year earlier). Laurie Werth replied within an hour: "congratulations!!" A few minutes later, Norman did the same: "This is great news. Thank you Don for all the hard work and encouragement." Werth then had second thoughts: "Yes, but is it Y2k-compliant?" Langford chimed in from England (adding below Norman's message): "Hear, hear…hard taskmaster as he undoubtedly was, Don nursed this project to success against considerable odds, and thoroughly deserves congratulations from us (and everyone else!)". The last to join the party (and perhaps the one who killed it with a touch of reality) was Mechler:

> Congratulations Don and condolences for the IEEE meeting.
> Congratulations to the whole team.
> Has it been determined where the code will reside?
> Maybe success has gone to my head but is there a next step we should consider?[79]

Amr El-Kadi emailed Gotterbarn directly two days later (October 22, 1998): "I[t] was exceptional news that we have reached this stage with Code, thanks to your leadership. Ed noted there were problems with IEEE, what happened there"?[80]

Too exhausted and exhilarated to write much, Gotterbarn had omitted some interesting details about what happened at the ACM Council meeting on October 19. He had taken time during his presentation in Vancouver to respond to the three issues the Council had raised in Washington in May. First, COPE had compared the ACM Code with the Software Engineering Code and found no incompatibility. Second, the two codes were not in competition. Those covered by the two codes overlapped but were not coextensive. Not all ACM members are software engineers; not all software engineers are ACM members. Nor was there anything odd about one person being subject to two codes of ethics. A Professional Engineer, for example, is subject to the code of ethics of both the National Society of Professional Engineers and that of his branch of engineering (such as the code of ethics of American Society of Mechanical Engineers). Third, COPE had recommended a four-stage approval process (which seems much more orderly as described here than it did as it developed):

> A. Develop and revise the Code based on extensive review [completed "July 1997"].
> B. Have the Code reviewed by the Joint ACM/IEEE-CS Task Force [this had included the Fall 1997 publication of Version 3 and ACM/IEEE-CS membership voting on it and resulted in the revisions that produced Version 4].
> C. Use an existing and tested formal review process used by the IEEE for approving international technical standards [Version 4 had easily passed that test but, as a result of suggestions made during the balloting, had been revised twice, generating Version 5.1].
> D. Formal adoption of [Version 5.1] by the ACM and Computer Society…as their approved code for the practice and teaching of software engineering.[81]

Though the ACM Council unanimously passed a motion appropriate to stage D (the last), it did not approve precisely the motion Gotterbarn expected. The Council approved the Code on condition an ACM "grammarian" examine it. At least one member of the Council was unhappy with some split infinitives.[82]

On November 20 (still, technically, a month before the last day of autumn), the IEEE-CS Board of Governors also voted to approve the Code (Version 5.1). But it too had a condition: its approval was subject to IEEE's lawyer examining the document and determining that approval would not cause IEEE-CS any legal problems.[83] On December 10, the lawyer, Sandra Pfau, answered the Board's concerns in a one-page memo to Anne Marie Kelly, Director, Volunteer Services, IEEE-CS (Woods' superior). Pfau stressed that standard setting is always legally risky, offering this chilling footnote:

> A recent example of the extensive liability associated with association standard setting is the case of National Spa & Pool Institute of Alexandria, Virginia. Due to a $6.6 million judgment against the Institute related to a man's life-changing injuries in a 1991 diving board incident, the institute has had to declare bankruptcy to keep operating while it appeals.

Voluntary standards are, however, "less problematic in the legal sense" than certification programs or "product standards". For this reason, "the language included in the code of ethics is perhaps not as significant as the manner in which the code is disseminated and publicized." The IEEE-CS should "make it clear when the code is published that it is a voluntary code." More

important, when the IEEE-CS endorses the Code, it should "make it clear that the endorsement goes to the use of the code, not to individuals or companies that claim to abide by the code." The distinction Pfau had in mind seems to be between endorsing the Code and certifying (or otherwise endorsing) specific practitioners or products. That is indeed an important distinction, but not one that divides "voluntary" from "involuntary" codes. Her next sentence (and the whole paragraph immediately below it) collapses the one distinction into the other. While the IEEE-CS "can say that it believes that following the code is a good thing, it is not a good idea…to say or imply that those who follow or have adopted the code are better software engineers than those that have not adopted the code." She was, of course, right that merely (formally) adopting the Code would not make one a better software engineer. But, unless everyone involved in writing the Code was badly mistaken, she must be wrong that *following* the Code does not make one a better software engineer. Of course, it does. That is why the IEEE-CS would—and should—say that "it is a good thing" to follow the Code.

The IEEE-CS Board of Governors bravely took this memo as favoring approval. The lawyer had, however, noted one "typo". Having corrected it, along with the faults the ACM grammarian had identified, Gotterbarn had before him Version 5.2 of the Software Engineering Code of Ethics and Professional Conduct, approved both by the ACM and the IEEE-CS (11.Appendix). It is hard to know exactly when this happened. But there is a memo that Gotterbarn sent Tripp on January 18, 1999, in which Gotterbarn both admitted his pleasure "that the Code has passed legal muster" and informed Tripp that "clause 6.13" had been corrected (as Pfau had suggested).[84] So, the process of approval was complete no later than January 18—and perhaps a few days before that, just under a month later than Tripp had estimated in May and just over five years after Gotterbarn had met with Barbacci and Melford in Pittsburgh to organize SEEPP. When the great moment came, it was an anticlimax.

11. Appendix**:**

# SOFTWARE ENGINEERING CODE OF ETHICS AND PROFESSIONAL PRACTICE

(Version 5.2) as recommended by the
IEEE-CS/ACM Joint Task Force on Software Engineering Ethics and Professional Practices and
Jointly approved by the ACM and the IEEE-CS as the standard for teaching and practicing software engineering.

Short Version

PREAMBLE
The short version of the code summarizes aspirations at a high level of abstraction. The clauses that are included in the full version give examples and details of how these aspirations change the way we act as software engineering professionals. Without the aspirations, the details can become legalistic and tedious; without the details, the aspirations can become high sounding but empty; together, the aspirations and the details form a cohesive code.

Software engineers shall commit themselves to making the analysis, specification, design, development, testing and maintenance of software a beneficial and respected profession. In accordance with their commitment to the health, safety and welfare of the public, software engineers shall adhere to the following Eight Principles:

1 PUBLIC - Software engineers shall act consistently with the public interest.

2 CLIENT AND EMPLOYER - Software engineers shall act in a manner that is in the best interests of their client and employer, consistent with the public interest.

3 PRODUCT - Software engineers shall ensure that their products and related modifications meet the highest professional standards possible.

4 JUDGMENT - Software engineers shall maintain integrity and independence in their professional judgment.

5 MANAGEMENT - Software engineering managers and leaders shall subscribe to and promote an ethical approach to the management of software development and maintenance.

6 PROFESSION - Software engineers shall advance the integrity and reputation of the profession consistent with the public interest.

7 COLLEAGUES - Software engineers shall be fair to and supportive of their colleagues.

8 SELF - Software engineers shall participate in lifelong learning regarding the practice of their

profession and shall promote an ethical approach to the practice of the profession.

**SOFTWARE ENGINEERING CODE OF ETHICS AND PROFESSIONAL PRACTICE**

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

Full Version

PREAMBLE

Computers have a central and growing role in commerce, industry, government, medicine, education, entertainment and society at large. Software engineers are those who contribute by direct participation or by teaching, to the analysis, specification, design, development, certification, maintenance and testing of software systems. Because of their roles in developing software systems, software engineers have significant opportunities to do good or cause harm, to enable others to do good or cause harm, or to influence others to do good or cause harm. To ensure, as much as possible, that their efforts will be used for good, software engineers must commit themselves to making software engineering a beneficial and respected profession. In accordance with that commitment, software engineers shall adhere to the following Code of Ethics and Professional Practice.

The Code contains eight Principles related to the behavior of and decisions made by professional software engineers, including practitioners, educators, managers, supervisors and policy makers, as well as trainees and students of the profession. The Principles identify the ethically responsible relationships in which individuals, groups, and organizations participate and the primary obligations within these relationships. The Clauses of each Principle are illustrations of some of the obligations included in these relationships. These obligations are founded in the software engineer's humanity, in special care owed to people affected by the work of software engineers, and in the unique elements of the practice of software engineering. The Code prescribes these as obligations of anyone claiming to be or aspiring to be a software engineer.

It is not intended that the individual parts of the Code be used in isolation to justify errors of omission or commission. The list of Principles and Clauses is not exhaustive. The Clauses 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 that generates ethical decisions. In some situations, standards may be in tension with each other or with standards from other sources. These situations require the software engineer to use ethical judgment to act in a manner which is most consistent with the spirit of the Code of Ethics and Professional Practice, given the circumstances.

Ethical tensions can best be addressed by thoughtful consideration of fundamental principles, rather than blind reliance on detailed regulations. These Principles should influence software engineers to consider broadly who is affected by their work; to examine if they and their colleagues are treating other human beings with due respect; to consider how the public, if reasonably well informed, would view their decisions; to analyze how the least empowered will be

affected by their decisions; and to consider whether their acts would be judged worthy of the ideal professional working as a software engineer. In all these judgments concern for the health, safety and welfare of the public is primary; that is, the "Public Interest" is central to this Code.

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 software engineers and managers of software engineers who need to take positive action in a specific case by documenting the ethical stance of the profession. The Code provides an ethical foundation to which individuals within teams and the team as a whole can appeal. The Code helps to define those actions that are ethically improper to request of a software engineer or teams of software engineers.

The Code is not simply for adjudicating the nature of questionable acts; it also has an important educational function. As this Code expresses the consensus of the profession on ethical issues, it is a means to educate both the public and aspiring professionals about the ethical obligations of all software engineers.

PRINCIPLES

Principle 1 PUBLIC Software engineers shall act consistently with the public interest. In particular, software engineers shall, as appropriate:

1.01. Accept full responsibility for their own work.
1.02. Moderate the interests of the software engineer, the employer, the client and the users with the public good.
1.03. Approve software only if they have a well-founded belief that it is safe, meets specifications, passes appropriate tests, and does not diminish quality of life, diminish privacy or harm the environment. The ultimate effect of the work should be to the public good.
1.04. Disclose to appropriate persons or authorities any actual or potential danger to the user, the public, or the environment, that they reasonably believe to be associated with software or related documents.
1.05. Cooperate in efforts to address matters of grave public concern caused by software, its installation, maintenance, support or documentation.
1.06. Be fair and avoid deception in all statements, particularly public ones, concerning software or related documents, methods and tools.
1.07. Consider issues of physical disabilities, allocation of resources, economic disadvantage and other factors that can diminish access to the benefits of software.
1.08. Be encouraged to volunteer professional skills to good causes and to contribute to public education concerning the discipline.

Principle 2 CLIENT AND EMPLOYER Software engineers shall act in a manner that is in the best interests of their client and employer, consistent with the public interest. In particular, software engineers shall, as appropriate:

2.01. Provide service in their areas of competence, being honest and forthright about any limitations of their experience and education.
2.02. Not knowingly use software that is obtained or retained either illegally or unethically.
2.03. Use the property of a client or employer only in ways properly authorized, and with the client's or employer's knowledge and consent.
2.04. Ensure that any document upon which they rely has been approved, when required, by someone authorized to approve it.
2.05. Keep private any confidential information gained in their professional work, where such confidentiality is consistent with the public interest and consistent with the law.
2.06. Identify, document, collect evidence and report to the client or the employer promptly if, in their opinion, a project is likely to fail, to prove too expensive, to violate intellectual property law, or otherwise to be problematic.
2.07. Identify, document, and report significant issues of social concern, of which they are aware, in software or related documents, to the employer or the client.
2.08. Accept no outside work detrimental to the work they perform for their primary employer.
2.09. Promote no interest adverse to their employer or client, unless a higher ethical concern is being compromised; in that case, inform the employer or another appropriate authority of the ethical concern.

Principle 3 PRODUCT Software engineers shall ensure that their products and related modifications meet the highest professional standards possible. In particular, software engineers shall, as appropriate:

3.01. Strive for high quality, acceptable cost, and a reasonable schedule, ensuring significant tradeoffs are clear to and accepted by the employer and the client, and are available for consideration by the user and the public.
3.02. Ensure proper and achievable goals and objectives for any project on which they work or propose.
3.03. Identify, define and address ethical, economic, cultural, legal and environmental issues related to work projects.
3.04. Ensure that they are qualified for any project on which they work or propose to work, by an appropriate combination of education, training, and experience,.
3.05. Ensure that an appropriate method is used for any project on which they work or propose to work.
3.06. Work to follow professional standards, when available, that are most appropriate for the task at hand, departing from these only when ethically or technically justified.
3.07. Strive to fully understand the specifications for software on which they work.
3.08. Ensure that specifications for software on which they work have been well documented, satisfy the users' requirements and have the appropriate approvals.
3.09. Ensure realistic quantitative estimates of cost, scheduling, personnel, quality and outcomes on any project on which they work or propose to work and provide an uncertainty assessment of these estimates.
3.10. Ensure adequate testing, debugging, and review of software and related documents on

which they work.
3.11. Ensure adequate documentation, including significant problems discovered and solutions adopted, for any project on which they work.
3.12. Work to develop software and related documents that respect the privacy of those who will be affected by that software.
3.13. Be careful to use only accurate data derived by ethical and lawful means, and use it only in ways properly authorized.
3.14. Maintain the integrity of data, being sensitive to outdated or flawed occurrences.
3.15 Treat all forms of software maintenance with the same professionalism as new development.

Principle 4 JUDGMENT Software engineers shall maintain integrity and independence in their professional judgment. In particular, software engineers shall, as appropriate:

4.01. Temper all technical judgments by the need to support and maintain human values.
4.02 Only endorse documents either prepared under their supervision or within their areas of competence and with which they are in agreement.
4.03. Maintain professional objectivity with respect to any software or related documents they are asked to evaluate.
4.04. Not engage in deceptive financial practices such as bribery, double billing, or other improper financial practices.
4.05. Disclose to all concerned parties those conflicts of interest that cannot reasonably be avoided or escaped.
4.06. Refuse to participate, as members or advisors, in a private, governmental or professional body concerned with software related issues, in which they, their employers or their clients have undisclosed potential conflicts of interest.

Principle 5 MANAGEMENT Software engineering managers and leaders shall subscribe to and promote an ethical approach to the management of software development and maintenance. In particular, those managing or leading software engineers shall, as appropriate:

5.01 Ensure good management for any project on which they work, including effective procedures for promotion of quality and reduction of risk.
5.02. Ensure that software engineers are informed of standards before being held to them.
5.03. Ensure that software engineers know the employer's policies and procedures for protecting passwords, files and information that is confidential to the employer or confidential to others.
5.04. Assign work only after taking into account appropriate contributions of education and experience tempered with a desire to further that education and experience.
5.05. Ensure realistic quantitative estimates of cost, scheduling, personnel, quality and outcomes on any project on which they work or propose to work, and provide an uncertainty assessment of these estimates.
5.06. Attract potential software engineers only by full and accurate description of the conditions of employment.

5.07. Offer fair and just remuneration.
5.08. Not unjustly prevent someone from taking a position for which that person is suitably qualified.
5.09. Ensure that there is a fair agreement concerning ownership of any software, processes, research, writing, or other intellectual property to which a software engineer has contributed.
5.10. Provide for due process in hearing charges of violation of an employer's policy or of this Code.
5.11. Not ask a software engineer to do anything inconsistent with this Code.
5.12. Not punish anyone for expressing ethical concerns about a project.

Principle 6 PROFESSION Software engineers shall advance the integrity and reputation of the profession consistent with the public interest. In particular, software engineers shall, as appropriate:

6.01. Help develop an organizational environment favorable to acting ethically.
6.02. Promote public knowledge of software engineering.
6.03. Extend software engineering knowledge by appropriate participation in professional organizations, meetings and publications.
6.04. Support, as members of a profession, other software engineers striving to follow this Code.
6.05. Not promote their own interest at the expense of the profession, client or employer.
6.06. Obey all laws governing their work, unless, in exceptional circumstances, such compliance is inconsistent with the public interest.
6.07. Be accurate in stating the characteristics of software on which they work, avoiding not only false claims but also claims that might reasonably be supposed to be speculative, vacuous, deceptive, misleading, or doubtful.
6.08. Take responsibility for detecting, correcting, and reporting errors in software and associated documents on which they work.
6.09. Ensure that clients, employers, and supervisors know of the software engineer's commitment to this Code of ethics, and the subsequent ramifications of such commitment.
6.10. Avoid associations with businesses and organizations which are in conflict with this code.
6.11. Recognize that violations of this Code are inconsistent with being a professional software engineer.
6.12. Express concerns to the people involved when significant violations of this Code are detected unless this is impossible, counter-productive, or dangerous.
6.13. Report significant violations of this Code to appropriate authorities when it is clear that consultation with people involved in these significant violations is impossible, counter-productive or dangerous.

Principle 7 COLLEAGUES Software engineers shall be fair to and supportive of their colleagues. In particular, software engineers shall, as appropriate:

7.01. Encourage colleagues to adhere to this Code.
7.02. Assist colleagues in professional development.

7.03. Credit fully the work of others and refrain from taking undue credit.
7.04. Review the work of others in an objective, candid, and properly-documented way.
7.05. Give a fair hearing to the opinions, concerns, or complaints of a colleague.
7.06. Assist colleagues in being fully aware of current standard work practices including policies and procedures for protecting passwords, files and other confidential information, and security measures in general.

7.07. Not unfairly intervene in the career of any colleague; however, concern for the employer, the client or public interest may compel software engineers, in good faith, to question the competence of a colleague.
7.08. 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 participate in lifelong learning regarding the practice of their profession and shall promote an ethical approach to the practice of the profession. In particular, software engineers shall continually endeavor to:

8.01. Further their knowledge of developments in the analysis, specification, design, development, maintenance 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 produce accurate, informative, and well-written documentation.
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 relevant standards and 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 Not give unfair treatment to anyone because of any irrelevant prejudices.
8.08. Not influence others to undertake any action that involves a breach of this Code.
8.09. Recognize that personal violations of this Code are inconsistent with being a professional software engineer.

This Code was developed by the IEEE-CS/ACM joint task force on Software Engineering Ethics and Professional Practices (SEEPP):
Executive Committee: Donald Gotterbarn (Chair),
Keith Miller and Simon Rogerson;
Members: Steve Barber, Peter Barnes, Ilene Burnstein, Michael Davis, Amr El-Kadi, N. Ben Fairweather, Milton Fulghum, N. Jayaram, Tom Jewett, Mark Kanko, Ernie Kallman, Duncan Langford, Joyce Currie Little, Ed Mechler, Manuel J. Norman, Douglas Phillips, Peter Ron Prinzivalli, Patrick Sullivan, John Weckert, Vivian Weil, S. Weisband and Laurie Honour Werth.

NOTES

[1] Of course, this "politically correct" and "American" language entered the Code when Gotterbarn was working in England under the watchful eye of that Scot-patriot Simon Rogerson. Gotterbarn was then trying to make the code "more international". It is remarkable how different from our intentions our achievements can be (or, at least, seem to others)!

[2] The Dutchman also mentioned (old) 5.06 ("Attract employees only by full and accurate description of the conditions of employment") in the same sentence, but apparently as an example of a standard already "enforced by local law". Gotterbarn\Survey Comments\ SLEGERS. Clause 5.05 had begun as 6.10 in Version 1; 5.06, as 6.12. The survey had given each a favorable rating just under 94%. Gotterbarn\Version 3\V3 Survey votes. Why had the two been deleted? Not all provisions receiving less than 95% had been deleted. Some, like 1.03 (the old 2.02), had received only 90% approval—and would continue to cause trouble. How did Gotterbarn decide what to revise and what to leave the same?

[3] Interview of Sigut, September 27, 2002. His education identifies him as an engineer (strictly so called): "I received a Diploma in Engineering in Mechanical Engineering in 1972 (roughly equivalent to an MS) from the Federal Institute of Technology (ETH for *E*idgenoessiche *T*echnische *H*ochschule). I began work as a structural engineer in 1972, using computers as part of the job. As time went on, writing software became an ever-bigger part of the job, until I came to think of myself as a software engineer. There was not then a career path leading to software engineering."

[4] Compare Sigut's comments in the July 20 vote (Gotterbarn\Version 5\RPLY):

> Second sentence of the third paragraph of the PREAMBLE to the Full Version says: 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. When we analyse this sentence, we can say about "the list" that it contains a. Principles b. Clauses ... and that it ... c. is not exhaustive d. should not be read as separating ... We therefore have two distinctive parts of the list, which can be said to have two properties each. Let us look at the four possible combinations: 1. The list of Clauses is not exhaustive OK, the list of Clauses can of course be continued infinitely. 2. The list of Clauses should not be read as ... OK, Necessary condition, otherwise inflexible. 3. The list of Principles is not exhaustive… the list of Principles IS the base of the Code. Of course, we might later find omissions, but that can be adjusted when we get there. If we say NOW that the list of Principles is not exhaustive, we better sit down and think again. 4. The list of Principles should not be read as ... It is not the Principles, but it is their application - embodied e.g. in the Clauses - , which can separate the acceptable from the unacceptable. It is not necessary to state that something impossible does not apply. Considering the four combinations we can see, that while the properties of the Clauses are well defined by the sentence, we can NOT say the same about the properties of the Principles. I therefore propose the sentence to be changed to read: The list of Clauses is not exhaustive, and should not be read as separating the acceptable from the unacceptable in professional conduct in all practical situations.

[5] Clause 7.06 survived unchanged through Version 5.2.

[6] Sigut Archive: 1-8-98. Sigut responded to Version 4 on January 12. That response is a bit over a page long. Most of his suggestions are minor (including several typos to fix), but he does "again…plead for" omitting "Principles and" because "while I agree with 'the list of Clauses' not being exhaustive, I have strong doubts and misgivings about saying the same about the Principles." This time (apparently) Gotterbarn did not respond. The objectionable language survives in Version 5.2.

[7] This provision is, of course, another one in need of editing: "on which…they…propose" is ungrammatical. 3.02 should read (something like): "Ensure proper and achievable goals and objectives for any project on which they work *or propose to work*" (or perhaps "Ensure proper and achievable goals and objectives for any project *which they propose or* on which they work"). This unfortunate sentence survives in Version 5.2.

[8] Zweben's letter is lost. We have only the response, Gotterbarn\Version 4\STURESP (no date on letter but file date is 1-19-1998). Apparently, Zweben had suggested "balance" instead of "moderate" (nearly an exact synonym). Gotterbarn did not explain how this weaker provision (in either form) was consistent with moving Principle "Public" from third place to first, with the claim in the Preamble that the public good is "primary", or with various Principles requiring that serving the client, employer, and profession should be done "consistent with the public interest". This language survive into Version 5.2.

[9] Indeed, except for its date (January 19), its first paragraph, the three sentences of the second paragraph, and the salutation ("Dear Steering Committee for the Professionalization of Software Engineering"), it is identical to the one sent December 16.

[10] Gotterbarn\Version 4\REC2 (or the identical: Gotterbarn\Steering Committee\ Recommendation of 4\REC2).

[11] Gotterbarn\Version 4\REC2.

[12] Gotterbarn\Version 4\Progress. The letter nonetheless carries "19 January 1998" as its date (and that is also the file date). Gotterbarn\Version 4\REC2.

[13] Gotterbarn\Version 4\Progress.

[14] Cabrera seems to have resigned from the Committee at the same time he resigned as chair. There was, it seems, no replacement.

[15] During Spring 1998, Gotterbarn used the title "co-chair" to describe the chair or vice-chair of the Steering Committee, apparently slipping into the language of his own early days working as one of SEEPP's two "co-chairs".

[16] The site www.computer.org/tab/seprof still existed on March 16, 2003 (without any obvious updating from five years before). Everything at that site, except Version 4, dated from March, 1997, another sign that the Steering Committee had been disintegrating for almost a year.

[17] According to Gerald Engel (then co-chair of the Curriculum Task Force), things were not as bad as Gotterbarn thought. ABET's request concerned accreditation criteria (something much less detailed than a curriculum). The work of the Curriculum Task Force would be (more or less) independent of what ABET was doing. Engel admits that this is a point likely to be lost on everyone but an expert in accreditation and curriculum.

[18] Charles (Chuck) House was then Intel's Science Policy and Society Impact Director. Before joining Intel in 1995, he had been President of Spectron MicroSystems, a wholly-owned subsidiary of Dialogic, focusing on distance learning and "corner conference room" enhancements. He had long been active in the IEEE as well as in the ACM (at one time serving as an IEEE Vice-President for Publications). www.cpd.ogi.edu/egm/house.htm (3/23/2004). He did not respond to email, classic mail, or phone messages requesting an interview.

[19] Gotterbarn\Version 4\PRESIDAC. I attribute the letter to Gotterbarn even though all three members of SEEPP's executive committee are listed as signatories. While not much turns on the attribution, I feel more comfortable treating Gotterbarn as sole author both because the letter uses "I" much more often than "we" (though it uses both) and because some of the detail (for example, events at a conference) seem personal. While I am sure that Gotterbarn cleared the letter with Miller and Rogerson, there is in fact no record even of that in the archives.

[20] Gotterbarn did not say it, but he knew about these forthcoming publications because they were his, part of the strategy of dissemination to prevent suppression. See: "The Uniqueness of Software Errors and Their Impact on Global Policy", *Science and Engineering Ethics* 4 (July 1998): 351-356; "Not all Codes are Created Equal: The Software Engineering Code of Ethics, a success story", *Journal of Business Ethics* 22 (October 1999): 81-89); and "The Ethical Computer Practitioner—Licensing the Moral Community: a proactive approach", *SIGCSE Bulletin* 30 (June 1998): 8-10. "The Proceedings of the ACM Policy Conference in May" were printed in a special issue of *Computer and Society* (with Keith Miller editing it along with Tom Jewett). Gotterbarn's contribution was: "Raising the bar: a software engineering code of ethics and professional practice", *Computers and Society* 28 (June 1998): 26-28 (with Version 4 in the next section). Gotterbarn was then vice-chair (and "Information Director") of the Special Interest Group (SIG) sponsoring that journal. If there was an ethics network within ACM, Gotterbarn was now at its center.

[21] Gotterbarn\Steering Committee\Dennis.

[22] Would the ACM Executive Council also notice that the short version said it was not to be severed from the long version because the short version's "aspirations can become high sounding but empty" without the details in the long version? Did Gotterbarn now silently thank Fairweather for insisting that something be said in the introduction to the short version to prevent its being severed from the long?

[23] Gotterbarn Chapter10cmt (September 30, 2004).

[24] www.computer.org/csinfo/bios/tripp.htm (3/16/2004).

[25] "[One] of the things we were warned about using as a model very early by Elliot C at a steering committee meeting was going through this process which normally takes YEARS because of infighting by vendors and review processes." Email (Gotterbarn to Davis), March 17, 2004. Compare Ch. 3.6.

[26] Gotterbarn\Version 4\Tripp1. There is no evidence that Tripp acknowledged receipt of this report.

[27] Mechler\SEEPP98.

[28] Gotterbarn\Version 4\ACM2 (with letterhead) and Gotterbarn\Version 4\ACM1 (without letterhead or addressee).

[29] Apparently, Gotterbarn communicated with Frailey again, by phone or email, after April 8. The archive does not contain that email (or any other to or from Frailey) before the May 10 meeting (or after).

[30] Gotterbarn\Version 4\Ethics Report (fwd). Carver—and her ACM co-chair—were both replaced at the end of 1996. The new co-chairs were Gerald Engel (IEEE-CS) and Rich LeBlanc (ACM). They served until the work of the task force was completed in 1998. www.acm.org/serving/se/Rpt9811.htm (3/23/2004).

[31] Item 3.2. Gotterbarn\Version 5\acmsupport\POL98MIN.

[32] For Frailey's overheads for that report, see www.acm.org/serving/se/min990128 (4/11/2004), attachment entitled "History of SWECC". Frailey titled this set of overheads the Steering Committtee's "Final Report". The overheads report two recommendations. Beside recommending "formal review, adoption, and approval processes be followed", it also recommended "continuing the code of ethics task force because of ongoing needs in professional practices area."

[33] For details of the report they delivered in October 1998, see 12.1 (and Gotterbarn\Version 5\ACMSUPPORT\ Report).

[34] The Steering Committee's share is even larger if we include item 3.6—an update on the integration of the CSAB with ABET. (There is a reference under that item to the Steering Committee's task force on Curriculum.) The motion in support of the CSAB-ABET Memo of Agreement passed without opposition but with three abstentions. For those wondering what "CSAB" stands for, the answer, according to Article I of its Constitution (October 15, 2000), is "CSAB Inc." (not, as widely supposed, "Computing Sciences Accreditation Board").

[35] The minutes attribute the substance of this paragraph to the "TF" (presumably, the "task force", not "Tripp and Frailey"). Gotterbarn thinks the minutes should have said "Steering Committee". Email (Gotterbarn to Davis), March 17, 2004. There are, however, at least two explanations of "TF" that would not reduce it to a mere recording error. One is that the Council members were as unclear about the distinction between the task force and Steering Committee as were many of us who participated in SEEPP's work. The other possibility (perhaps the more likely) is that the recording secretary was simply echoing the language of the Steering Committee motion—and that (like the letter Tripp was to write two weeks later), it attributed the request for a formal review to SEEPP (rather than to the Steering Committee).

[36] "Council is being asked to formally adopt this Code of Ethics per Leonard Tripp's…letter. Gotterbarn suggested that having a long delay for some minor changes to the Code is not advisable."

[37] www.acm.org/key.people/ar/cope98 ((3/19/2004).

[38] Email (Gotterbarn to Davis), March 17, 2004.

[39] Gotterbarn does not recall the exact date, but it was probably May 14 (Thursday) because, on that date, both Gotterbarn and Tripp seem to have opened their first files concerned with balloting—at about 11:34 AM (PDT). See, for example, Gotterbarn\Version 4 IEEE Vote\aBallot_ethics. Why did Gotterbarn not contact Tripp earlier that week, say, late morning or early afternoon, Monday, May 11? That would seem to be the most likely time for Gotterbarn to call Tripp, the first working day after Gotterbarn's meeting with the ACM Executive Council. Gotterbarn now had every reason to move quickly. The only reason for him to wait even till mid-day on Monday to call was that he, in EDT, would have been calling Seattle, in PDT. Of course, that he called on May 11 is one thing. That he actually reached Tripp right away is another. There may have been several days of "telephone tag" (as with Carver), ending on Thursday morning (May 14). That would explain why nothing happened until May 14—and then why a lot happened very quickly.

[40] Email (Gotterbarn to Davis), March 21, 2004 (attachment).

[41] Email (Gotterbarn to Davis), March 21, 2004 (attachment).

[42] Gotterbarn\Version 4 IEEE Vote\Ballot. The originals of the other two seem not to have survived. Since Gotterbarn had begun to use attachments, when he saved a revised version, he would have wiped out its predecessor (while preserving original file information).

[43] Gotterbarn\Version 4 IEEE Vote\ABALRSP1.

[44] Gotterbarn\Version 4 IEEE Vote\aInvitcmt.

[45] Gotterbarn\Version 4 IEEE Vote\AAVOTERS. This is an intermediate list. What seems to be the final list is: Gotterbarn\Version 4 IEEE Vote\AAVOTOUT. The covering memo (addressed to "Leonard") reads:

> Here is a text file of addresses of possible balloters. It includes names from several email lists related to software engineering and some related to ethics. It also includes the ethics officers from some Multinationals, e.g. Higgins from Boeing. Twenty six of the emails are for the SEEPP task force members.

This file was, it seems, opened on May 14, about 3:45 PM (and so must have been revised after the May 15th email in which Gotterbarn asked advice. (Yes, Burnstein, Weil, and I are on this list—with the right email addresses.)

[46] This was, of course, technically inaccurate. Neither SEEPP as a whole, nor its executive committee, had recommended a formal review process; it was the Steering Committee that had done that, with Gotterbarn eventually going along. What SEEPP (or, rather, its executive committee) had recommended was that the executive bodies of the two societies approve the Code without delay.

[47] I point out the use of the personal pronoun "we" here because that is its only use in a document that otherwise seems to speak in the name of the "IEEE and ACM".

[48] IIT's archive seems to have stopped receiving messages through the listserv sometime after December 18, 1997. This does not seem to have been caused by any change of address. I was receiving email from Mechler as late as July 7, 1998, at csep@charlie.acc.iit.edu (and Gotterbarn's May 14, 1998 list gives the correct addresses for everyone at IIT).

[49] Donald Gotterbarn, "Two Computer-Related Codes", *Perspectives on the Professions* 19 (Fall 1999): 4-6 The whole issue was about writing codes of ethics.

[50] Mechler\SEEPP98.

[51] Email (Gotterbarn to Davis), March 17, 2004. Gotterbarn added, "I did not consider the headers useful so I deleted them." So, we do not know even how many emails we should look for.

[52] Mechler\SEEPP98. There are at least three of these emails: two on 06/05/98; and one on 06/15/98.

[53] Compare Gotterbarn: "For all I know 100 people from Microsoft could have wanted to ballot but only two, that I know of, were on the official balloting group. (I don't know the rules or how it is done but they are very scrupulous to have a broad group.) They applied the same techniques for ballot group selection as when they are dealing with major technical standards like a communications protocol." Email (Gotterbarn to Davis), March 21, 2004 (attachment). For Gotterbarn, the IEEE Standards process is a blackbox.

[54] Mechler\SEEPP98. Gotterbarn then went back to the subject of what Mechler might have missed concerning SEEPP, repeating information he had already given Mechler several weeks

before: “I am working with the president elect of ieee-cs [Tripp] to follow their procedures for doing the ballots—following out their formal process.”

[55] The address problem was not “fixed”. On July 10, Mechler again wrote Gotterbarn about it (in an otherwise joking note): “I voted yesterday, guess which way? Attached is a WORD 6.0 of the short version [of the Code?] suitable for framing, vertically or horizontally with some changes in margins. It fits nice in a 8x11 frame in my office. If you want you can pass it around, I am still having trouble with my e-mail address.” Mechler\SEEPP98.

[56] Mechler\SEEPP98. “Gotterbarn\History of SE Code\History expanded” gives the date for the end of balloting as July 15 (perhaps relying on Tripp’s original schedule rather than the final one).

[57] Or, perhaps, the number should be 146. Two balloters (Samaras, #12 and #24, and Swearingen #114 and #135) each appear twice in the list of 148. Gotterbarn\1998\IEEE vote replies. But the covering memo also lists “Robert J. (Rob) Schaff” (sic) as having voted twice. He is in fact not listed even once among the 148 (though “Schaaf” does appear as #75 and #76). So, the number actually voting might be 145. Given this range, and because the exact percentages do not matter, I will use the IEEE numbers even though they use 148 as the base.

[58] I reached the low of 7% by going through the balloters names, looking for any that appeared on any of Gotterbarn’s lists (not only of earlier survey respondents but of the original SEEPP working group members, correspondents, and so on). I found eleven names (out of a possible 148). I found four SEEPP members who might have voted on Version 3 (three of whom certainly did): Burnstein, Little, Mechler, and Langford. I also found six others (not SEEPP members) who voted on Version 3 or at least commented on it: Berleur, Black , Harding, LaRue, Maner, and Sigut). (That means that ten people—just under 7%—certainly did vote both times.) I also recognized three other names not on any of Gotterbarn’s SEEPP lists: Joe Herkert (an engineer who has become interested in engineering ethics); Judith Perrolle (whom Gotterbarn knew, if from no where else, from working on the ACM code of ethics); and Rob Riser (Gotterbarn’s colleague at ETSU and on COPE). Because Gotterbarn might have urged them to vote, they too might have voted on Version 3 (bringing the total to 13, almost 9%). Since there were about thirty who voted on Version 3 but had no comment (or, at least, no comment that survives with a name attached), including Herkert, Perrolle, and Riser, it is possible that the overlap could be as high as 40 (10 + 30) names (about 27%).

[59] I have inserted “should have” because we have evidence that at least two voters seem not to have realized that comments would be public. One gave his confidential ACM voter number. Another, after voting for the Code, mentioned in his comments that he did not find the Code helpful in dealing with an ethical problem he faced:

> I have recently agreed to serve as treasurer for my church. I have been assured that the position involves overseeing assistants that are well qualified to perform the clerical parts of the job. In particular, the pastor’s wife will do the payroll on bootleg software.

Gotterbarn had both the ACM number and this story removed from the comments before posting them on the IEEE-CS web site. (I have achieved a similar protection of confidentiality by not saying which of the 148 was the source of the story.) Gotterbarn\1998\IEEE Vote Replies.

[60] Gotterbarn\1998\IEEE Vote Replies (#5, #11, #48, #52). We have the earlier comments from Berleur, Sigut, and Black, but know of Harding's only because he referred to them in his ballot comments. For Berleur's, see Chapter 8.7 (including notes); for Sigut's, 11.1. Black's comments on Version 3, fourteen pages long, are in Gotterbarn\Version 3\Comment\Black42. Based on them alone, I would not have counted Black as an (overall) negative on Version 3 (just a careful reader, thoughtful critic, and devoted supporter of a code of some sort). I count him as an overall negative on Version 3 only because he does not indicate, in his explanation of his negative vote on Version 4, that any change from Version 3 affected his vote. The negative vote seems to have been his way of making sure Gotterbarn at least considered one particular point he had made about Version 3 as well as many new comments concerning Version 4. As he said in his Interview, October 11, 2002:

> I sent in comments on V. 3 after seeing that blurb. I also commented on V. 4 after receiving an invitation to do so from Donald Gotterbarn. Most of my comments were on presentation, for example, why use the word "third party" in a few places when the text would be clearer if it used one or more of the terms already standard throughout the code ("public", "user", etc.)? Only a few of my comments were substantive. In a few places, I thought they had gone beyond what was appropriate for a professional code, for example, by calling upon software engineers to volunteer for civic work [in Clause 1.08]. I liked the code overall, especially, it saying that it was intended as "guidelines" to help you think through the problems you'll face.

Black does not sound like a "true" negative. Nor did Gotterbarn so consider him. The version of Black's comments on 3 that Gotterbarn preserved has this header (in bold): "THESE COMMENTS ARE WORTH A REVISIT ! DG".

[61] Unfortunately, I have only memory to testify to the *Queen Mary* simile. Mechler's Interview, June 11, 2002, contains only the following exchange (under Question 8) that makes the same point in a less colorful way:

> **MD:** Did you participate in any of the other versions?
>
> **EM:** I saw them, but I didn't have any problems with them. I sent them around to the committees. Really, the only time anyone complained was when the condensed version was put out, and that was really the only thing. At that time, and I don't know how anyone else felt, but we felt that we did our job.

[62] Interview of Jayaram, February 25, 2003 (under Question 17).

[63] The oddly named Junk was not alone in making this objection. Gotterbarn did not, however, follow Junk's straightforward suggestion. Version 5.2 now reads: "1.03 Approve software only if they have a well-founded belief that it is safe, meets specifications, passes appropriate tests, and does not diminish quality of life, diminish privacy or harm the environment. The ultimate effect of the work should be to the public good." The sentence tacked on the end (in violation of the general format of the clauses) seems to be Gotterbarn's response to Junk. But it does not seem to me to resolve it (unless one is a utilitarian). There might, it seems, be times when we should not diminish quality of life, diminish privacy, or harm the environment even when, all things considered, the public would benefit (a little). We should not do that, for example, when the small public benefit is achieved at the cost of substantial harm to a few individuals or the environment.

[64] Gotterbarn eventually had 185 distinct responses. See email (Gotterbarn to Davis), March 16, 2004 (attachment), apparently, an email to Tripp, giving SEEPP's history through the end of 1998. The number of distinct responses is given at 8/31 (the end of the formal review process).

[65] SEEPP's Guide to Operations 4.3.5 (apparently following 1993 IEEE Standards Operations Manual) set the percentage necessary for passage at 75%, much closer to Version 4's 77% but still on the right side of the line. The Guide also set 75% as the minimum return rate for the balloting to be valid (while by 1998 the IEEE was using 60% for the minimum return rate). Gotterbarn\94-96 Misc\OPGuide feb96.

[66] While at work on Version 5, Gotterbarn received a cheering email from Prinzivalli:

> Has there been any plans developed or considered about how the Societies (IEEE/CS & ACM) are going to disseminate the Code? I have some ideas about this state of the project and am willing to work with others of the same ilk to develop the follow-on training; e.g. strctures of class room or workshop sessions, Code training session-leader, and shared omnibudsman tasks for the Code's Publication and Implementation Plan.

Gotterbarn\Version 5\The code and beyond.

[67] Gotterbarn\History of SE Code\History expanded: "8/12/98 v 5 to Tripp". There is a file for Version 5.0 with that date on it (Gotterbarn\History of SE Code\ SEEPPV5toTrip). The corresponding file for comments is dated "8/13/1998" (Gotterbarn\History of SE Code\REPLY). SERPLYDN (with file date 8/12/1998), though virtually identical to REPLY, seems to be an intermediate document of some sort.

[68] Gotterbarn\1998\IEEE vote responses.

[69] Gotterbarn\Version 5\Leonard IEE [sic] Vote Code of Ethics.

[70] Not many of the working documents for Version 5.1 survive. The only ones I have found are: Gotterbarn\ by\1998\Version 4 after IEEE vote\reply Aug 13; and (apparently identical) Gotterbarn\ by\1998\IEEE Formal Review\Jacques Berleur committees comments.

[71] Or, perhaps it went to everyone who had initially been invited to vote, whether they voted or not. What suggests this possibility is a “NOTE” near the end of the letter: “A person who did not respond to initial ballot may not raise objections to any part of the draft code of ethics which has not been changed from the initial ballot.” They may, then, still object to something new—even though they did not even vote Abstain the first time.

[72] While this email continues to calculate percentages for the original vote using 148 as the base, it uses 146 (not 145) to calculate the percentage of resolved and unresolved negatives—and that final 86.3% affirmative (which would be 85.1% using the 148 base).

[73] Sigut was not at all happy with this webpage. On September 11, he wrote Woods (in part):

> The quality of the "Comment Disposition Report" page is so substandard, that I am ashamed to refer anybody to it. If you want to see what it could look like, check "http://www.awu.id.ethz.ch/~sigut/Comment.html". In that document, I changed my Response (nr.11) so, that it corresponds to the original form, and the spurious errors (0A, 0D, !, ...) are deleted. As a side-effect, the file is ca. 3'320 bytes SHORTER, but that's another story.

Sigut saved the webpage and, following his interview, sent me the file (From Sigut—Software Engineering Code of Ethics Comments.doc). The file consists of a brief report and the spreadsheet. The report is clear, but the (crucial) spreadsheet is virtually unreadable (though one can make it readable by copying to a blank document and playing with it there). Sigut’s file did not include the ballot below the spreadsheet, though he had a problem with that too: “I do not see, how I can ‘complete the questionnaire on the ... Web page’, it being a plain text.” Apparently, even the IEEE has (or, at least, had) trouble integrating software and making it do what it wanted.

[74] The website displayed the following documents: Recirculation Ballot; Code of Ethics (Version 5.0); Summary of Changes; Comment Disposition Report; Ballot Group Member List; and Questionnaire. The Summary of Changes has survived (Gotterbarn\Version 5\Changes) as has the Comment Disposition Report in something less than its final form (Gotterbarn\Version 4 IEEE vote\sevote5a\SEVOTErpl). Among the changes the summary reported were a great many concerned with “grammar” (for example, changing “judgement” to “judgment” [undoing the spelling introduced while Gotterbarn worked in England], “consistent” to “consistently”, “methodology” to “method”, and “literate” to “well-written”). Among the substantive changes was the addition of “management of maintenance” to Principle 5, of “maintenance and support” to 1.05; the addition of “privacy” in 1.03; the substitution of “uncertainty” for “risk” in 3.09 and 5.05; the substitution of “Be encouraged to volunteer” for “Volunteer” in 1.08; and the addition of a “non-discrimination” clause (“8.07 Not give unfair treatment to anyone because of any irrelevant prejudices”). The difference between Version 4 and 5.0, though significant, is tiny

compared to that between Version 1 and 2, 2 and 3, or 3 and 4. The IEEE Standards process was bringing the drafting process to a close.

[75] The number had actually been 33, with Schaaf having voted twice. He was now listed once. The number should, however, have been 31 because G.M. Samaras is still listed twice (unless he is distinct from George M. Samaras). I should add (for those familiar with machine politics) that there was confusion at IEEE but no hanky-panky. G.M. changed his vote; George did not vote this time. Gotterbarn\Version 4 IEEE Vote\Counts.

[76] Reading these negative votes is not as easy as it may seem. When I interviewed Sigut (September 27, 2002), he seemed to like the Code *on the whole*. When I asked what he didn't like, he produced this list ("based on version 5.2 of the Code"):

> Full version; PREAMBLE; 1st paragraph; 2nd sentence: The text "are those who" could be left out. As it is, the sentence can be interpreted as a definition of a "software engineer" instead of (as I understand it) an enumeration of the software engineers' contributions.
>
> Full Version; PREAMBLE; 3rd paragraph; 2nd sentence: It still says "The list of Principles [...] is not exhaustive". Apart from my critique, I understood during the interview that this is NOT the intended meaning.
>
> Looking at the Clauses 1.04, 6.06 and 6.13 I ask: "who will protect the Software Engineer who reports or disobeys the law"? If the backing is not guaranteed, acting according to these clauses (or the given exception) might lead to unpleasant results.
>
> Clause 1.02: I would have chosen "temper" instead of "moderate".
>
> Clause 2.01: I would suggest using "frank" or "candid" instead of "forthright".

[77] Mechler\SEEPP98.

[78] Mechler\SEEPP98. Version 5.1 differed from Version 5.0 only in "some minor editorial" corrections. Gotterbarn\1998\ACM Council Response. A Word document comparison revealed the following differences: "client or employer" in Principle 2 has become "client and employer"; two sentences in the first paragraph of the Preamble have been revised a bit; the word "quantitative" has been inserted after "realistic" in 3.09; "outcome" in 5.05 has been pluralized; "better" has been deleted from 5.09 ("taking a better position"); the credits have been reorganized; and (perhaps most important) a copyright in the name of the "SEEPP Executive Committee" has been added.

[79] Mechler\SEEPP98.

[80] Gotterbarn\version 5\October 23 Passed.

[81] Gotterbarn\1998\ACM Council Response.

[82] Email (Gotterbarn to Davis), April 2, 2004. Comparing Version 5.1 with 5.2 suggests that the grammarian ignored the split infinitives (such as that in 3.07) but suggested changing "consistent" to "consistently" in Principle 1, adding a "shall" after the "and" in Principle 8, changing "consider" to "recognize" in 6.11 and 8.09, and other minor changes of the same sort. The grammarian definitely did not perform the review for consistency and style I had recommended the year before.

[83] On November 30, Mechler emailed Gotterbarn, "Did IEEE [accept] the code? You said they would look at it in November." An hour later Gotterbarn answered, "ieee-cs approved the code 'subject to legal review'. I thought I would wait on announcing to the group until the legal eagle said it was ok. More later." Mechler\SEEPP98.

[84] Email (Gotterbarn to Davis), March 16, 2004 (attachment). Actually, the date on the memo is "1/18/98" (rather than "99"), an error common soon after the change of year. The January memo also includes Gotterbarn's first attempt at a timeline for SEEPP's five year history.

# Chapter 12: End Game, Version 5.2, 1999-2000

"It isn't that things will necessarily go wrong (Murphy's Law), but rather that they will take so much more time and effort than you think if they are not to go wrong."
—Wolf's Law

## 12.1 The Joint Steering Committee's happy end

On November 20, 1998, the Joint Steering Committee presented its "annual report" to the IEEE-CS Board of Governors. Prepared by "Dennis Frailey, co-chair and Leonard Tripp, chair", the report began by describing five "highlights" of an evaluation of the Committee's operations "for the period 1993 to 1997". (The evaluation had been undertaken soon after Tripp took over six months before.)[1] These "highlights" (under the heading "Committee Reorganization") seem to represent the thinking of the Committee, or at least of Tripp and Frailey, early in 1998. The first "highlight" was that the IEEE-CS and ACM should "continue the work begun by the steering committee." What the Committee was doing was worthwhile; it only needed to be done in a better way. Second, the Body of Knowledge task force would require "professional staff support" to finish its work. The work had proved too much for a task force consisting entirely of volunteers (and specialists they could hire under grants they were able to obtain). The third evaluation should be familiar (from 11.2): the "current draft of the Code of Ethics" (Version 4) is ready "for review in a formal consensus process with a balanced representation in the review team". Fourth, there was no reason for the Curriculum task force to wait for the Body of Knowledge task force to complete its work. The preparation of "curriculum criteria is ready to commence". The last "valuation" was a recommendation: that the ACM and IEEE-CS jointly sponsor a "permanent entity" to carry on the work of the Steering Committee (and, by implication, that the Steering Committee be disbanded).

This section of the report ends with the announcement that a "charter for the new committee" was prepared in 1998, approved by the ACM Executive Council and the IEEE-CS Board of Governors, and put into operation. The first "Executive Committee" of this new entity—the "Software Engineering Coordinating Committee" (SWECC)—had six members (listed in alphabetical order). Three of the names are new: Mark Ardis (Bell Labs), Linda Northrop (Software Engineering Institute), and Karl Reed (Department of Computer Science and Computer Engineering, La Trobe University, Australia). The other three are familiar: Carver, Failey, and Tripp. Of the six, only one (Frailey) is from the original Joint Steering Committee (Carver having been a member only *ex officio* for the Committee's first two years). According to a working document (September 29, 1998), Frailey was to be "Vice-Chair", representing the "ACM EC"; Ardis, to be the ACM member representing the "technical community" (and endorsed by SIGSOFT); and Northrop, to be the ACM member representing the "education community" (endorsed by the "ACM Ed Board Chair").[2] Those labels suggest that the other three are the IEEE-CS members: Carver (IEEE-CS President) representing the IEEE-CS Board of Governors; Reed (a professor), representing the "educational community"; and Tripp (at Boeing), "the technical community". In fact, when they were listed in the minutes of SWECC's first meeting, Tripp was only identified as "Chair". Carver (rather than Reed) was identified as

"CS Education" (that is, the representative of the "education community") and Reed (rather than Tripp) as "CS TCSE" (the representative of the "technical community").[3]

In its fifty-fifty split between ACM and IEEE-CS appointees, SWECC's executive committee is plainly the Steering Committee's successor. It is also the Steering Committee's successor in having under it "project committees" that, at first glance, seem to be "task forces" under a new name.[4] In several other respects, however, SWECC is a different entity. First, it was to have only six members, roughly half the number of the old (ten-member) Joint Steering Committee (excluding *ex officios*), but exactly the number the IEEE-CS Steering Committee had before the ACM joined late in 1993. Apparently, ten had proved too large for the purpose. Second, though SWECC may have as many project committees as required, it could create them only with "the consent of the Executive Committee of the ACM and the Executive Committee of the [IEEE-]CS". Each project would have "an approved duration".[5] The project would go out of existence after that date—unless the ACM and IEEE-CS authorized an extension. Projects were to find it harder than task forces to go on year after year. Third, the chair was to have a two-year term and rotate between the two societies. The first chair was to be from the IEEE-CS (and to be appointed by its Board of Governors); the next, appointed by the ACM Executive Council (from among SWECC members); the third, appointed in the same way by the Board; and so on. The ACM and IEEE-CS were to be exactly equal in SWECC. Indeed, this equality was to extend to the projects. When SWECC "formed a project", it was to create a "committee for its oversight with equal membership from the ACM and the CS." SWECC could, it seems, avoid the disastrous double-chairing that had nearly sunk SEEPP, but it could (it may seem) never again allow an executive committee like that Gotterbarn set up in January 1997. The next Gotterbarn would have to choose an executive committee with an equal number of ACM and IEEE-CS members; the next Patricia Douglas would have to find an ACM co-chair right away; and so on.[6]

We should not be too quick to treat this prospective elimination of the "next Gotterbarn" as a condemnation of the actual Gotterbarn—or as nearly as important a constraint as it sounds. In fact, the overlap between ACM and IEEE-CS allowed Gotterbarn to keep his executive committee just as it was when (as we shall see in12.3) SEEPP became "SEPEP" (one of SWECC's projects). Gotterbarn could be a member for both "ACM and IEEE-CS", Miller, for IEEE-CS alone (though he was also a member of ACM); and Rogerson, a member of ACM (and the British Computer Society) but not of IEEE-CS, could counter-balance Miller. That preserved equality between the two societies—if only technically.[7] The ease with which the executive committee of SEEPP's successor satisfied the new requirement is a reminder that no one seems to have objected to how Gotterbarn organized SEEPP. Indeed, his handling of SEEPP earned him a "retroactive promotion". Early in January 1999, Tripp asked Gotterbarn to provide a timeline for SEEPP. Gotterbarn did as asked, explicitly mentioning his co-chair, Robert Melford, at appropriate moments.[8] Yet, when the Steering Committee published its "History" (in March 1999), there was no mention of Melford. The history lists Gotterbarn as "chair" for the entire period, 1994-98 (as if there had been no reorganization at the end of 1996). What makes the lack of any mention of Gotterbarn's co-char seem significant is that the History lists all the other task force chairs (all as "co-chairs"), even those who (like Carver and John Werth) had been out of office for as long as Gotterbarn's co-chair.[9]

SWECC differed from the Joint Steering Committee in another way worth mention. The purpose of the Steering Committee had been to *establish* software engineering as a profession. That purpose at least suggested that software engineering was not yet a profession. In fact, when

the ACM joined the IEEE-CS Steering Committee, it had expressed doubts about whether having a profession of software engineering was even a good idea (2.5), apparently embodying those doubts in the appointment of Mary Shaw as one of its representatives. SWECC's charter suggests that the general understanding of software engineering may have changed between 1993 and 1998. SWECC's purpose, according to the charter, was "to foster and maintain software engineering as a professional computing discipline." While "foster" still expresses doubt that software engineering is *fully* a "professional computing discipline", "maintain" leaves no doubt that it is at least *in part* such a profession (as well as, in part, a "discipline" within "computing"). When the ACM Executive Council approved SWECC's charter (and did not appoint Shaw to the Steering Committee's successor), it in effect declared that Shaw had lost the argument over the *professional* "maturity" of software engineering.[10]

The Annual Report listed six "Key Events" for 1998, the first two of which occurred in June. The ACM and IEEE-CS agreed to work with the Texas Board of Professional Engineers to "identify a technical basis for recognizing competency in software engineering" and "[enact] rules that recognized software engineering as a distinct engineering discipline." In August, the NCEES (National Council of Examiners for Engineering and Surveying) passed a resolution in support of "licensing software engineering throughout the nation." These first three events, so closely bunched, may help to explain why Shaw had lost the argument (for the moment at least). Whatever the ACM or IEEE-CS thought, events seemed to be quickly turning software engineering into a part of engineering. The only question still open seemed to be how it would be done. The next three "key events" offered a tentative answer. In September, the Industry Advisory Board for the Guide to the Software Body of Knowledge Project "held its first meeting in Mont Tremblant, Quebec". The body of knowledge task force (soon to be known as "SWEBOK") was hard at work on the "Guide" that licensing bodies would need if they were to offer examinations in software engineering. In October, the engineering accreditation agency (ABET) and its computer-science counterpart (CSAB) "announced the integration of their accreditation services". And, on November 11, the ACM and IEEE-CS jointly hosted a meeting to "initiate the update of Computer Curricula 1991". Led by software engineering, computer science seemed to be moving closer to engineering.[11]

The Steering Committee's last report ended with two "Project Results" in keeping with these events. The first was that the ACM and IEEE-CS had approved the software engineering code of ethics. (Like the Guide to the Body of Knowledge, the Code provided a basis for distinguishing between those who should be licensed as software engineers and those who should not.) The second "project result" was that the Curriculum task force had completed its "model accreditation criteria for software engineering" in November and that the ACM and IEEE-CS "education boards" had endorsed them. Though the Curriculum task force had started much more slowly than SEEPP, it had finished at almost the same time. When the Steering Committee officially dissolved at the end of 1998, it could take considerable satisfaction in what it had accomplished. Of its four original assignments, it had completed two, defining "ethical standards" and "educational curricula" (or, at least, standards for such curricula). A third (defining the body of knowledge) was well on its way to completion (though it had proved a much bigger job than expected). Only the fourth, "adopting standard definitions", had been abandoned—and only because the Steering Committee had decided that the IEEE already provided an adequate definition of "software engineering" and related terms.[12] The Joint Steering Committee might (like General Wolfe on the Plains of Abraham) "die content".

## 12.2 SWECC takes over

On January 28, 1999 (in Austin, Texas), SWECC held its first meeting, a nine-hour marathon beginning at 8:30 AM, with only twenty minutes for lunch. All six members were present (though Northrop was only present by a proxy, Laurie Werth, a resident of Austin).[13] The minutes, nine single-spaced pages, are full of interesting summaries of past work and plans for the future. The minutes have six attachments: "History of SWECC"; "Overview of Software Engineering Standards"; "Report on SWEBOK Project"; "Overview presentation given to Texas Board of Prof. Engineers"; "Overview of SWECC"; and a "White paper by Dennis Frailey on cooperation". (All but the last attachment consist of several pages of overheads.) All these documents (as well as the Steering Committee's Annual Report) were placed on the web. (Tripp seems to have liked the sun to shine on whatever he was doing.)

We may draw three conclusions from an examination of these early SWECC documents. The first is that Frailey and Tripp, but especially Tripp, seem to be the source of most of the activity. For example, two of the attachments are Frailey's ("History" and "White paper"); the other four seem to be Tripp's. The minutes identify five "Responsibility Areas". Frailey has one ("Education"); Reed and Northrop share one ("Publicity"); and the other three are Tripp's ("SWEBOK", "Liaison with Standards", and "Liaison with National Bodies").

The second conclusion to draw from the minutes of January 28 is that the Body of Knowledge is not only Tripp's focus but also SWECC's (just as it had been for the Joint Steering Committee) and, as they believed, integral to the Curriculum project (now "Education"). Much of the morning half of the Austin meeting seems to have been informational (with the first session devoted to "Overview and History of SWECC Project" and the second to "Relationships between SWECC and others). The last session of the morning is, like the first three afternoon sessions (each an academic hour), for "Plans". But then there is a short session (twenty-five minutes) on "Software Engineering Standards" followed by a long session (fifty-five minutes) for "Report on SWEBOK Project"—with "Responsibility Areas" and "Future Plans" completing the day. There was, it seems, no special session for Education, even though all three highlighted resolutions of the day explicitly mentioned the "Education Project". SWECC adopted three resolutions at the Austin meeting. Two of the three mention Education along with SWEBOK. The third, while it does not explicitly mention SWEBOK, at least implicitly relies on its results: the undergraduate curriculum is to be the "priority" for the Education Project and that curriculum is to include "the core content of the software engineering field."

This description of what SWECC did at its first meeting suggests the third conclusion I draw from the minutes. The Code of Ethics was not on the agenda at all. It was, however, not entirely forgotten. It is mentioned once in the minutes—the last item on a list of four "proposed projects" (the other three of which are "Guide to SWEBOK", "SWE Model Curriculum", and "SWE Performance Norms").[14] The Code of Ethics was also mentioned in several of the overheads. Whatever the Code of Ethics was, it was no longer "a problem".

The next face-to-face meeting was to be in Montreal a half year later (July 13, 1999), apparently because the technical experts working on the Body of Knowledge were in Montreal (at the University of Quebec). SWECC also scheduled a "teleconference during the week of

April 12, 1999." It was, it seems, to be a short meeting, allowing SWECC to evaluate progress on various "action items".

## 12.3 Back in Tennessee

Meanwhile, Gotterbarn was trying to put the Code into the hands of as many software engineers as possible. These efforts took three forms: publication, consultation, and organization. Typical of the first was the publication in the January 1999 issue of *Software Engineering Notes* (an ACM SIGSOFT journal) of Version 5.2 along with a brief introduction under the title "A Positive Step Toward a Profession: The Software Engineering Code of Ethics and Professional Practice". Though the article was typical of Gotterbarn's attempts to publicize the Code, response to it was not. A letter to SEN's Editor objected both that the Code was "too vague" and that it added "nothing to the existing codes of ethics for engineers". The time had come, it asserted, "to move beyond 'motherhood statements' and try to give meaning to the title, 'Software Engineer', by establishing standards that restrict the use of the title to a proper (and probably small) subset of those who now use it."[15] The Editor notified Gotterbarn of the letter as soon as he received it, offering to print a response beside it in the next issue (May). For Gotterbarn, what began as a simple article had turned into a crisis. The letter's author, David Parnas, was (and remains) one of the "greats" of software engineering, especially among software engineers concerned with professional ethics.

The importance of Parnas to software engineering was evident in the very issue of SEN in which his letter to the editor appeared. A few pages after the letter is an "ACM Fellow Profile" of him, including both a short biography and a three-page interview. A licensed Professional Engineer in the Province of Ontario, with a BS, MS, and Ph.D. from Carnegie Mellon (all in Electrical Engineering), Parnas held the NSERC/Bell Industrial Research Chair in Software Engineering at McMasters University. He was the author of more than two hundred papers and reports, recipient of the ACM Best Paper Award in 1979, of two Most Influential Paper awards from the International Conference on Software Engineering (ICSE), and (in 1998) of the ACM SIGSOFT Outstanding Research Award. Those interested in software ethics revered him as one of the early opponents of Reagan's plan to defend against nuclear attack from space ("Star Wars"). Parnas resigned from a federal advisory panel in protest because "as a professional…I must devote some of my energy to deciding whether the task that I have been given is of benefit to society". He argued that the Star Wars project could not do what it was supposed to do, could not be adequately tested even if it could, and would in any case not be reliable enough to be worth the expense. He made his arguments in public, stressing that his resignation was not simply a "personal" decision but one any responsible professional in similar circumstances should take.[16]

Gotterbarn's response to Parnas' letter was four times as long as the letter itself. It was authored not by Gotterbarn alone, but by Miller and Rogerson as well, with "SEEPP Executive Committee" (rather than their academic affiliations) under their names (even though neither SEEPP nor its executive committee any longer existed).[17] Apparently, Gotterbarn thought he needed to speak with as much authority as possible. He also had to treat Parnas with respect, while explaining how wrong he was. The response justifies the level of abstraction the code adopted (what Parnas calls "vagueness") by arguing that the Code is just abstract enough not to use standards that are (and are likely to continue to be) "in flux". A code of ethics should not, for

example, require “mutation testing for all mission critical software”, though it should require “adequate testing” (however a software engineer should understand that term at the time). A code of ethics should be abstract enough not to be outdated in a few years.

A code of ethics should not, however, be so abstract as to be altogether “uncontroversial” —and the Software Engineering Code is demonstrably not that abstract. Here the IEEE Standards process provided evidence of how much controversy there was about the Code:

> In 149 pages of comments developed during the IEEE-CS formal review, the following conflicting comments were made by practicing “professional” software developers: add the clause “avoid criticism of software engineers’, “remove 1.08 …volunteer professional skills to good causes”, “…make 1.08 mandatory”, “remove references to maintenance”, “include all forms of maintenance”, “…not all jobs require the highest professional standards” [and so on].[18]

Gotterbarn attributes this variety of opinion to “software engineering [being], at best, an emerging profession”. That, I think, is a mistake. Diversity of opinion on such questions is, as far as I can tell, characteristic of all professions, even of law and medicine, and has always been, perhaps because any substantial number of human beings tend to have a wide range of opinion, unless forced to reach agreement by circumstance. What distinguishes the “fully developed professions” from those that are merely “emerging” is simply a core on which most (but not all) agree. In this respect at least, Gotterbarn’s data seems to show that software engineering was, by 1999, no longer an emerging profession. The Standards process had documented something close to ninety percent agreement on the Code as a whole (and *no* opposition to *most* provisions).

Gotterbarn might have added that the error of considering a code of ethics “uncontroversial”, “trivial”, or otherwise no more than common sense when it is not, is an “honorable error”, one most likely to be made by those who hold themselves to high standards. Assuming that others think as they do, they cannot see the point of the code’s requiring no more than they require of themselves. Those who work to a lower standard tend to make the opposite mistake, that is, to suppose that the code is “unrealistic” or “impractical”, sets too high a standard”, and so on. They are often surprised to discover that few of their colleagues agree with them about what is practical. The open discussion of what is “uncontroversial” and “impractical” itself helps to shape the discipline’s understanding of both. We have no direct insight into much of practice beyond our own and perhaps that of a few people with whom we work closely.

Below Gotterbarn’s response to Parnas was a related news item. The ACM had established an Advisory Panel on Professional Licensing in Software Engineering to assess “the current role of ACM as part of a joint effort with the IEEE Computer Society to oversee and actively participate in the development of professionalism and licensing guidelines through [SWECC]”, to recommend “an appropriate role for ACM to take with respect to these issues”, and to provide a “brief final report to the ACM Council by its next meeting [May 16. 1999]”. Among the fourteen members of this advisory panel was David Parnas.[19] While the committee was “free to determine its course”, the Council expected it to “consult with the representatives of the areas of ACM that have an interest in the topic [including]: SIGSOFT (David Notkin, Chair), the Ed Board (Peter Denning, Chair), and the ACM representatives to SWECC (Dennis Frailey [and the others]).” Licensing was again causing trouble, more than Gotterbarn could then guess.

The Gotterbarn-Parnas exchange is just one example of how Gotterbarn sought to disseminate the Code through publication.[20] Typical of attempts to disseminate it through consulting is an exchange of emails with Dell Computer's legal department ("Dell Legal"). Late in January, a paralegal at Dell Legal wrote Gotterbarn. That email is lost, but Gotterbarn's January 22 response makes it clear that he understood Dell to be asking permission to copy the Code. He informed the paralegal, "The copyright is registered to me", expressed pleasure at Dell's interest, and asked what exactly Dell had in mind.[21] A few days later, the paralegal explained that her "team is working to enhance our existing guidelines for procurement professionals to be appropriate for a wide group within Dell". Her team would like to adopt some of the Code's "concepts" but it "would not be a direct adoption". She then offered to send Gotterbarn a draft if "it would make you more comfortable". The draft of "1/26/99" ("Standards of Business Ethics for PSG Professionals/Engineering Code of Conduct") arrived a few days later (February 16). Nine pages long, it was an impressive document, but showed no obvious borrowing from the Code.

Gotterbarn's statement that the Code's copyright is "registered to me" deserves comment. Actually, the copyright was in the name of Miller and Rogerson as well as Gotterbarn. The copyright in question seems to have covered all versions of the code, starting with "version 1.0".[22] Copyrighting the Code was part of a strategy, dating from September 18, 1998, to avoid two dangers. One was that the short version of the Code might be printed without the long. The other danger was that the ACM or IEEE-CS might charge for the Code (as they did for other documents), thus (unintentionally) discouraging dissemination and use, or bury the Code by failing to grant permission to copy it. By copyrighting the Code, Gotterbarn avoided both dangers. He could provide free downloads of the Code at the SEERI website on condition that the Code not be divided and that no charge be made for copies. Neither the ACM nor IEEE-CS was happy with this arrangement, however. They viewed the Code as a "work for hire" and therefore theirs, not Gotterbarn's, even though no one who had had any part in writing the code had been hired (except the ACM grammarian and IEEE-CS lawyer). By the middle of 1999, Gotterbarn, ACM, and IEEE-CS had resolved the dispute. The two societies became joint holders of the copyright, but only after agreeing that the Code should carry below the copyright symbol the statement: "This Code may be published without permission as long as it is not changed in any way and it carries the copyright notice." It carries that notice to this day.[23]

Copyrighting the Code was a small part of efforts to create the organizational infrastructure to disseminate the Code more widely. The Software Engineering Ethics Research Institute (SEERI) was another (10. 2); so too was ACM's Committee on Professional Ethics (COPE). But once SWECC replaced the Joint Steering Committee, it seemed obvious that dissemination had to include SWECC, especially if SWECC got the funding and staff support it was asking for. Gotterbarn therefore developed a plan for a "project" under SWECC corresponding to the task force he had chaired under the Steering Committee. Unfortunately, by this time, the Software Engineering Education Project had the acronym "SWEEP". So, Gotterbarn could not name the project he was planning "the Software Engineering Ethics Project". Its obvious acronym, SEEP (not to mention "SWEEP"), would be too easy to confuse with the Education Project's SWEEP. Gotterbarn dismissed "the Software Engineering Ethics and Professional Practices Project" (SEEPPP or SWEEPPP) for the same reason. Eventually, he chose "SEPEP" (Software Engineering Professionalism and Ethics Project), with the lucky suggestion of energy ("PEP"). Consistency with the nomenclature of other SWECC names

would entail a "W" after the "S", but Gotterbarn thought that consistency not worth the loss of the imperative sounding "SE[E]".

Gotterbarn submitted a plan to SWECC's meeting on April 15. SEPEP's plan had four parts: Purpose, Structure and Governance, Immediate Projects, and Initial Committee Membership.[24] SEPEP's purposes included publicizing the Code, fostering broader acceptance of the Code across international borders, updating the Code "as appropriate", and working with SWEEP to incorporate into software engineering education a knowledge of the Code and its use in decision making. Among SEPEP's immediate projects were developing a flyer about the Code to be included with all ACM and IEEE-CS membership renewals, presenting a panel on the Code at appropriate ACM and IEEE-CS software engineering meetings, publishing the Code in *Communications of the ACM* and *Computer*, working with the Education group to develop educational materials on the Code, and maintaining a joint ACM/IEEE-CS website on the Code. SEPEP was to have a three-member executive committee. Its chair was to be "a member of both the ACM and the Computer Society", with the other two members representing (respectively) the ACM and the IEEE-CS. The chair would report to SWECC annually. Though SWECC (in January) had required all projects to have a definite duration, SEPEP's plan included no termination date (though it did provide for a fiscal year beginning July 1). Nonetheless, the plan was approved "as presented".[25]

The meeting at which this happened (a teleconference) lasted only seventy-five minutes. SEPEP had fifteen of those minutes. Ahead of SEPEP was SWEEP's plan. The chair (Tripp) presented it "[on] behalf of Jerry Engel and Richard LeBlanc". SWEEP had forty-five minutes. After SEPEP was "new business", a fifteen-minute discussion of "opportunities for promoting the Guide to the Software Body of Knowledge Project". All the "action items" for the meeting concerned the Body of Knowledge. All six members were "present".

## 12.4 A happy but short life

When SWECC next met (Montreal, July 13, 1999), only four members were present (Ardis, Carver, Frailey, and Tripp). At another all-day meeting, they approved "Guidelines for Projects and Proposals for New Projects", the minutes of the January and April meetings, and a proposal to conduct annual surveys of international software engineering programs. They also approved the content of a "Code of Ethics poster", though with the suggestion that "the presentation could be improved by consulting with a graphics artist". Perhaps most important, however, SWEBOK reported that the "Stoneman" version of the Guide to the Software Body of Knowledge (version 0.5, that is, something well short of Version 1.0) was nearly complete, with the remaining two Knowledge Areas expected to be complete by July 23. A "few hundred" reviewers had volunteered, covering all the Knowledge Areas, but more reviewers were being sought "to represent other points of view". Beside the Texas Board of Professional Engineers, "other groups (especially in Canada) have been taking actions about licensing software engineers". There was agreement that SWECC "needs to enter into formal agreement for the exchange of materials" with all of these groups. SWECC would meet again in person in late January 2000 and, if there was enough business, by teleconference before then, in October.

SWECC did not meet again, even by teleconference, until March 4, 2000. But it nonetheless ended 1999 with a major achievement, a whole issue of *IEEE Software* (November/December) devoted to it, almost sixty pages of text. Tripp (along with Steve

McConnell) was Guest Editor (with an Introduction to the issue, pp. 13-18). David Parnas contributed the lead article (pp. 19-30), "Software Engineering Programs Are Not Computer Science Programs", a spirited and intelligent justification for training software engineers as if software engineering were just another kind of engineering. Parnas even set out—course by course—the curriculum he would prescribe for an undergraduate software engineering student. Immediately following Parnas was Gerald Engel's report on SWEEP, "Program Criteria for Software Engineering Accreditation Programs" (pp. 31-34). While SWEEP's undergraduate criteria were designed to augment the computer science criteria ("CASC/CSAB"), the graduate criteria tracked ABET's engineering criteria. Following Engel's paper was "The Guide to the Software Engineering Body of Knowledge" (pp. 35-44). Three of its five authors (Piere Bourque, Robert Dupuis, and Alain Abran) were from the University of Quebec at Montreal. The other two were James Moore (MITRE Corporation) and Tripp. The Stoneman version of the Guide was available on the web for comment. Other versions would follow, each taking into account comments on the one before. The Body of Knowledge would always be a work in progress. Moore had an article of his own about another SWECC project, developing "An Integrated Collection of Software Engineering Standards" (pp. 51-57). Gotterbarn followed Moore, explaining "How the New Software Engineering Code of Ethics Affects You" (pp. 58-64). This article includes two inserts: one, the short version of the Code (without the long); the other, a list of "Early Adopters". Among the early adopters is McConnell's Construx Software (and the United Kingdom's Royal Mail).

The remaining three pieces concern licensing. The first of these ("What Do You Mean I Can't Call Myself a Software Engineer?") was in the middle of the issue (pp. 45-50). Its author, John Speed, was at the time a member of the Texas Board of Professional Engineers (and himself a licensed—civil—engineer). His article is largely descriptive, explaining what the new Texas regulations of the term "software engineer" require. The other two pieces (and a short "Responds", at the end of the issue, are officially polemical (being grouped under the title "Head to Head"). Dennis Frailey defended licensing (pp. 66 and 68). Tom DeMarco, a principal in a "New York and London based consulting group", argued against. For DeMarco, the market provides "the most efficient and elegant certification mechanism ever envisioned" (pp. 67 and 69). The only alternative to the market, a "Soviet-style central licensing bureaucracy", would do only one thing, protect "a class of senior practitioners whose skills are so irrelevant and so far out of date that they can't really do any highly paid job except licensing." Frailey's response to this free-market attack on licensing (p. 69) is to point out that DeMarco does not know what he is talking about. Licensing professionals is almost universal, even in countries without a "Soviet-style bureaucracy". And because licensing regulations allow the licensing body to impose conditions for keeping a license, "licensing is one of the few mechanisms that encourages and, in some jurisdictions, even requires continuing education." Licensing raises standards beyond what the market would otherwise impose.

Licensing was still an issue likely to rouse passions, perhaps as likely to rouse passions as it was in 1993 when Buckley included licensing in his IEEE-CS Board of Governors motion to begin establishing software engineering as a profession (2.3). Evidence of how licensing could still arouse passions appeared in the February 2000 issue of the *Communications of the ACM*. The ACM President, Barbara Simons, wrote a "Viewpoint" piece, "Not Now, Not like This", stressing that a licensing test "will [not] assure that the person who passes the test will be qualified to write programs that will never endanger the public." Part of the reason a test cannot

provide that assurance is that we "do not have building codes for programs." Indeed, we do not even have "a vocabulary of program design rich enough to discuss structural integrity."[26] Simons had two co-authors, Fran Allen and Paula Hawthorn, both members of the ACM advisory panel established in March 1999 to report on licensing software engineers.[27]

That panel had reported to the ACM Executive Council on May 15, 1999 (on schedule).[28] The Viewpoint piece was, in effect, disseminating the four conclusions of that report—which the Council had endorsed. The Council had declared that the "ACM opposed" licensing software engineers "at this time" because it believed that licensing would be "premature" and would not address "problems of software quality and reliability". The ACM was nonetheless "committed to solving the software quality problems", but (the Council declared) the way to do that was "by promoting R&D, by developing a core body of knowledge for software engineers, and by identifying standards of practice." While the ACM would continue to advise "boards and other agencies interested in licensing of software engineers", its "interaction will not imply ACM's endorsement of licensing or involve associated materials such as tests." Last, the Council undertook to form an "ACM SWECC Oversight Committee" to oversee "the development of the software engineering body of knowledge and ACM's participation in SWECC." This report *seemed* to favor most of what SWECC was doing.

The first and last appendices to the May 15 report provided further insight into how contentious licensing remained. Appendix A, by John Speed, is the official report of the Texas Board of Professional Engineers (approved February 4, 1999). A crucial element of the Texas approach was that a licensed "software engineer" had to be an engineer (strictly so called): "Not everyone with computer related experience is eligible to become a licensed Texas P.E. by applying under the software engineering discipline." The Board would decide "on a case-by-case basis who has been practicing engineering". This requirement was not as hard for computer scientists to satisfy as it might seem. The Board had a definition of "the practice of software engineering" that made explicit reference to "computer sciences". More likely to be discriminatory was the requirement that an applicant have letters of reference from "at least 9 people, 5 of whom must be licensed engineers."

Did computer scientists actually have anything to fear from engineers? Appendix C of the Report at least suggested that they did. According to that document, the Senate of the Memorial University of New Foundland, Memorial's academic governing board, had approved an honors program in the Department of Computer Science with a specialization in "software engineering". Upon learning of this degree, the Association of Professional Engineers and Geoscientists of New Foundland had joined the Canadian Council of Professional Engineers (CCPE) to *sue* Memorial over the use of the term "engineering". The case was set for hearing in the federal court in St. John's in September 1999.[29] The Association of Universities and Colleges of Canada (representing most of the degree-granting institutions in Canada) pledged to cover eighty percent of any legal costs Memorial incurred. What began as a simple academic decision had, because of the word "engineering", become an expensive federal case.[30]

On March 4, 2000, SWECC (again) met in Austin.[31] Northrop and Carver were absent, but the other four members were present, along with Gotterbarn (for the "Ethics Project"), Laurie Werth ("observer from Education Project"), Dupuis and Bourgue ("for SWEBOK"), and Gerald Engel (present for "the first part of the meeting to report on the progress of the [Education Project]" through his representative, Laurie Werth). The "Summary of Resolutions" gives the impression that the Ethics Project was the central concern of this meeting. Of four resolutions,

three concerned ethics. SWECC (now written "SWEcc") approved a logo for the code of ethics, approved copyrighting the logo as a "committee asset", and approved "the proposed revised budget for the code of ethics project and flyer distribution".[32] Another resolution listed in the Summary thanked "the SWEBOK team for the quality of its work and its contribution to software engineering as a profession." Though these resolutions together suggest a meeting both dull and short, the minutes themselves (four pages, with one page-long appendix of "Action Items" and another of "Acronyms") suggest something else.

The first important item of the day concerned proposals in Indiana, New Mexico, and Missouri to license software engineers. The proposal from Missouri was "particularly important" (so important that each SWECC member received a copy). It would treat software engineering as a specialty within computer engineering. The proposal would be submitted to the NCEES (National Council of Examiners for Engineering and Surveying) in Fall 2000 as an alternative to the Texas proposal. There was a "general perception that the Missouri proposal would be less desirable than the Texas proposal" not only because it could "divide the [computing] community but also [because it placed] software engineering in a less influential and fundamentally inappropriate position as a subsidiary of computer engineering in the minds of the professional engineering community." After considering possible responses, SWECC took no action. SWECC responded in the same way to the Texas Board's request to "review the definition of software engineering".[33]

After this half-page discussion, the minutes of SWECC's March meeting offered Gotterbarn's report on the Code of Ethics Project, a half-page report on an informal survey of software engineering programs, an informal report on SWEEP (Education), and a half-page report on SWEBOK (Body of Knowledge). Since July 1999, the Guide to SWEBOK had completed its "stoneman" phase and, despite "some relatively minor schedule slips due to delayed responses by certain authors", was close to completing the "ironman". SWECC planned no further face-to-face meetings in 2000. It would, however, have at least one conference call with SWEEP and SWEBOK "to review progress" and another conference call (even later) to "discuss approval of the SWEBOK final document".

That March meeting was SWECC's last. Three months later (June 30, 2000), the ACM's Executive Council voted to withdraw from SWECC, in effect killing the organization.[34] SWECC had survived little more than eighteen months, less than a third as long as the temporary Joint Steering Committee for which it was supposed to be the permanent replacement. The Executive Council that voted to withdraw from SWECC was the same one that had approved entry. It withdrew just before leaving office. To make clear that it did not take its decision lightly, the Executive Council issued a six-page explanation ("A Summary of the ACM Position on Software Engineering as a Licensed Engineering Profession"), with three appendices: A) two pages of "Q&A" summarizing the Report; B) a ten-page "Assessment of the Software Engineering Body of Knowledge Efforts"; and C) a twenty-page report "On Licensing of Software Engineers Working on Safety-Critical Software".[35] Two of the three authors of the Body of Knowledge report should be familiar: David Notkin *and* Mary Shaw.[36]

SWECC (according to the Summary) had "increasingly been perceived as furthering efforts to license software engineers." Two particular efforts stood out: "1) communications with the State of Texas relative to Texas' desire to develop a licensing exam for software engineers under the general framework for professional engineers (PE's); and 2) the development of a first-draft of the body of knowledge (SWEBOK) that all software engineers would be expected to

know." It was such efforts that led the Executive Council to appoint a "blue ribbon panel of prominent software engineers" to investigate "the issue of software engineering and the need for licensing". The Council had decided in May 1999 that "it could not support licensing software engineers." The state of knowledge and practice in software engineering is "too immature to warrant licensing". Licensing would, under the circumstances, be "ineffective in providing assurances about software quality and reliability." In May 2000, the Council concluded as well that "the framework of a licensed professional engineer, originally developed for civil engineers, does not match the professional industrial practice of software engineering." Because SWECC had become "so closely identified with licensing of software engineers under a professional engineer model, the ACM Council decided to withdraw from SWECC." Though the ACM had withdrawn from SWECC, it continued to believe that "the problem of reliable and dependable software, especially in critical applications, is the most important problem facing the IT profession." ACM would continue to "work closely with [IEEE-CS] on projects to further the evolution of software engineering as a professional computing discipline and improve the quality of software and the capabilities of software engineers."

There are, I think, two (compatible) ways to read what ACM did. One is that ACM had no objection to software engineering becoming a profession; the objection was to software engineering becoming an integral part of engineering (rather than a "professional computing discipline" or an "IT profession").[37] Integration into engineering would have important consequences for computer science as an academic enterprise. What consequences? Early in 1999, the Naval Academy had announced a graduate program in software engineering within Computer Science (and Information Systems), the first in the US.[38] Later that year, Carnegie Mellon University had announced a similar program. The CMU announcement listed a faculty of four, all from the School of Computer Science (Institute for Software Research International).One of the four was Mary Shaw.[39] The graduates of such programs might well have trouble getting licensed as software engineers in Texas. If the Texas approach spread, programs like that at the Naval Academy and CMU might die—and computer science programs be the poorer in consequence.[40]

The other way to read the ACM decision is as a misunderstanding. That was how Frailey read it. Writing for himself in the *Forum for Software Engineering Education* (FASE), he "strongly disapprove[d]" of the ACM decision; it would have "detrimental long term ramifications to software engineering and to the ACM"; and it put "in doubt" the work of "over 400 volunteers from about 50 countries" for "weak and often incorrect" reasons.[41] Relationships "with key external organizations" that had taken eight years to build had been severed with the loss of good will and respect for the ACM. Frailey was equally disappointed with how the ACM had reached its decision. The process had been "exclusionary". Neither the Executive Council nor its task forces had "consider[ed] the views and insights of ACM's appointed SWEcc representatives…the people actually doing the work—even after we offered to provide information and to assist in the discussions". One result was that the draft rationale contained "incorrect assumptions and factual errors" that, though easily corrected, "may well have influenced the Council's decision." For example, the draft report said that SWECC "focused…on matters of licensing", it did not—"ask anyone who has attended one of our meetings". SWECC had not, as suggested, supported licensing in ways inconsistent with ACM policy. In fact, it had "dissuaded the Texas Board from developing an examination and convinced them that ACM was an appropriate authority to turn to for advice in this matter." While Frailey hoped that "the work

of the four SWEcc projects will continue under some banner" supported by the "software engineering community", the work would now have to go on without "the coordination and communication functions that SWEcc provided, the united front presented by the ACM and the Computer Society working together, and the influence on these activities that ACM could have had." The ACM had, in effect, ceded the project of establishing software engineering as a profession to IEEE-CS—and IEEE-CS had always looked on licensing with more favor than the ACM had. For Frailey, ACM's withdrawal from SWECC was a strategic blunder.

## 12.5 Software Engineering Ethics Coordination after SWECC (2000-2004)

This book's importance depends (in part at least) on the importance of the process studied. If writing the Software Engineering Code in particular, or establishing software engineering as a profession in general, is merely a failed attempt to establish a profession, the thinking about professions preserved here will seem less important for understanding professions. This book would seem likely to tell us no more about living professions than an autopsy of a fetus spontaneously aborted can tell us about children born alive. We are, of course, not yet in position to know that software engineering has succeeded in becoming a profession. We need a decade or two—at least. Events between SWECC's dissolution and completion of this book do, however, provide some evidence that the attempt is succeeding. I can think of no better way to conclude this chapter than by (briefly) recounting that evidence.

Software engineering has continued its transformation from a field of computer science into a field of engineering (strictly so called). Two events seem especially relevant here. On March 14, 2001 (after what he described as "2+ years" of silence), Gotterbarn emailed a score or so of those involved in writing the SE Code of Ethics that (among other things) the "IEEE has started a Certification for Software Engineering Professional." In addition to "experience, education and passing a test they require adherence to the Code".[42] When I looked at the IEEE-CS website in 2004, I found only a certificate program for "Software Development Professional", but it did include a link to the SE Code of Ethics.[43] The Code remains an integral part of IEEE certification of software development professionals (if not exactly of software engineers), apparently the only IEEE-CS (or IEEE) certificate program. Whether or not certification is essential to profession, it is clear that the certification of software engineers is now treated as part of engineering certification—and linked to the SE Code.

Sometime in August 2001, the ACM's Task Force on Licensing of Software Engineers Working on Safety-Critical Software issued a "Final Report". The Report opposed licensing (for reasons already familiar) but endorsed "codes of professional conduct" (without mention of the Software Engineering Code by name).[44] The Task Force did, however, recommend that the ACM review its codes of ethics from time to time. While the ACM does not seem to have taken this recommendation to heart, the IEEE-CS has.

On November 7, 2002, the IEEE-CS Professional Practices Committee (Leonard Tripp, Chair) concluded that "the Code is stable and currently there is not sufficient evidence that it is in need of revision."[45] The Committee reported only two criticisms of the Code. One, apparently widespread, was that the Code did not "outline specific sanctions for violations". The Committee thought this no more than a misunderstanding: "Those sanctions are generally contained in the bylaws." The Code need only, and does, "refer to potential sanctions…[in] 8.09" ("Recognize that personal violations of this Code are inconsistent with being a professional software

engineer"). The other criticism received was not at all common: "One Florida based computer manufacturing company's lawyer said the company could not adopt the Code because Clause 6.06 admits the possibility of a circumstance where someone may ['in exceptional circumstances'] have to violate a law to protect the public." The company adopted all the other clauses of the code. The Committee thought the Code was right to say that the law should not trump the public interest.

The report's appendix listed some evidence of the Code's success: five textbooks in "Software Engineering" that either reprinted the Code or posted it on the book's website; nine textbooks in "Computer Ethics" that made similar use; seven companies that had adopted the code (and several more that posted it on their website); and adoption of the (short version) of the Code by the Association of Information Technology Professionals.[46] Most impressive, perhaps, were the number of translations posted on the web, three "official" (Chinese, Italian, and Spanish) and three more "unofficial" (Croatian, Hebrew, and Japanese). An official French translation was underway (for French Canadians).[47] (A translation becomes official when Gotterbarn, after consultation with software engineers who speak both languages, concludes that the translation is reasonably faithful to the original.)

On April 10, 2004, Keith Miller emailed Simon Rogerson (with copies to Gotterbarn and me) reporting that he had recently attended a workshop (in California) on software testing: "I was pleased to hear several speakers mention the joint ACM/IEEE code at the workshop when the issue of professionalism came up." Though the "lack of enforcement rules was mentioned", it was also "mentioned that the code was an important first step for the profession." The "technical world" seemed to know about the code and that knowledge seemed to "make a difference".[48]

Meanwhile, even the Guide to the Software Engineering Body of Knowledge (SWEBOK)—apart from licensing, the most controversial part of the professionalization project—continued to develop. On September 15, 2001, Dupius, Bourque, and Abram, the three Canadians who had taken over "editing" SWEBOK, issued an interim report. "Trial Version 0.95" (just short of Version 1) had been published in February 2001 for review. In April 2001, the project's Industrial Advisory Board "recognize[d] that due process was followed in the development of the Guide (Trial Version) and endorse[d] the position that the Guide (Trial Version) is ready for field trials for a period of two years." In May 2001, the IEEE-CS Board of Governors accepted SWEBOK (Trial Version) "as fulfilling its development requirements" and also declared it ready for two years of field trials.[49]

A little over a year later (July 25, 2002), Bourque emailed a general call for position papers and participation in a workshop on "improvements to [SWEBOK] and to the Software Engineering Education Body of Knowledge (SEEK) to be held October 6, 2002 at his home institution, the Ècole de technologie surpierieure, in Montreal. The results of the workshop were, in turn, to be used as "input to a second (to be confirmed) workshop" at the Conference on Software Engineering and Training (to be held in Madrid in March 2003). While SWEBOK now belonged entirely to IEEE-CS, SEEK was still an "ACM/IEEE Computer Society initiative to define undergraduate software engineering curriculum recommendations." On May 29, 2003, Bourque issued a call for volunteers to review Version 1.0—almost the final step for publication as a Technical Report by the International Organization for Standardization (ISO TR 19759). On February 6, 2004, the IEEE-CS Industrial Advisory Board unanimously agreed that "the Software Engineering Body of Knowledge project initiated in 1998 has been successfully completed; and endorse[d] the 2004 Version of the Guide to the SWEBOK and commend[ed] it

to the IEEE Computer Society Board of Governors for their approval." The IEEE-CS Board then approved the "2004 Edition of the Guide" (and authorized the Chair of the Professional Practices Committee to proceed with printing).[50] SWEBOK will hereafter be reviewed (and revised) like any other IEEE standard.

Software engineering now has an (internationally defined) body of knowledge, accreditation guidelines (both graduate and undergraduate), and a code of ethics. It has procedures in place to maintain (and improve) the relevant documents, to teach them to the next generation of software engineers, and even to integrate them into current practice. Assuming these achievements are neither mirages nor ephemera, what more does software engineering have to do to count as a profession (strictly so called)?

NOTES

[1] Annual Report – 1998, Joint IEEE Computer Society and ACM Steering Committee for the Establishment of Software Engineering as a Profession. www.acm.org/serving/se/Rpt9811 (3/23/2004). This is, as far as I can tell, the only annual report the Joint Committee ever issued (or, at least, ever placed on the web). The web version seems to have first appeared in November 1998 ("9811") and was last updated on "2 March 1999", but (according to the draft SWECC schedule of September 29, 1998) should have been prepared in February 1999. (Gotterbarn\Version 5\ACMSUPPORT\SCHEDV20). Since Frailey's name precedes Tripp's, the order of names is unusual. Usually, the chair's name would be listed first. While Tripp might be listed second simply in submission to the alphabet, the more likely explanation seems to be that the order recognized that Frailey did more of the work.

[2] Gotterbarn\Version 5\ACMSUPPORT\ACMREPS (file date: 9/29/98). Ardis is now a professor in the Department of Computer Science and Software Engineering, Rose-Hulman Institute of Technology, suggesting (as many other biographies here have as well) how porous is the boundary dividing software engineering's "technical community" from its "educational community".

[3] www.acm.org/serving/se/mi990128 (4/11/2004). By January 1999, Tripp had replaced Carver as IEEE-CS President. The people are, it seems, much more stable than what they "represent".

[4] Gotterbarn\Version 5\ACMSUPPORT\CHARV22 (file date: 9/29/98). The document itself indicates that it is as "approved on 19 October 1998" by the ACM, suggesting that it might be more recent. But I doubt it. The document also claims to be as approved by the IEEE-CS "XX November 1998". The final document (available at www.acm.org/serving/se/Charter, 4/11/2004) differs from it in a number of (minor) ways. My guess is that, on September 29, Frailey and Tripp knew the exact date of the ACM Executive Council meeting (only three weeks away) but not that of the IEEE-CS Board (almost two months off). They had merely "penciled in" the future. What we have here is very much a draft.

[5] Here is one place where the final Charter differs from the one in Gotterbarn's files. The language of Gotterbarn's draft was "an intended duration".

[6] SWECC's charter is also interesting for what it has to say about resources. The two societies agree to provide funding for SWECC, dividing expenses evenly, except that IEEE-CS is to "provide a secretariat service for SWECC". www.acm.org/serving/se/Charter (4/11/2004). Another draft document of September 28, 1998 is a budget for "FY 1999 and 2000", calling for a total of $12,600 for the first year and $25,000 for the second. Gotterbarn\Version 5\ ACMSUPPORT\BUDGET2. There is no budget on line.

[7] "Software Engineering Professionalism and Ethics Project (SEPEP), April 1999". www.acm.org/serving/se/Sepec (3/23/2004).

[8] Gotterbarn/History of SE code/ History of proj.

[9] www.acm.org/service/se/History (2/23/2004).

[10] Compare the "Rationale" for SWECC offered to ACM and IEEE-CS. Gotterbarn\Version 5\ACMSUPPORT\Report.

[11] ABET has long included "technology" programs as well as engineering programs in its accreditation work and, from the ACM's perspective, integrating CSAB with ABET may have seemed no more than a more efficient way to do what it had been doing all along.

[12] www.acm.org/service/se/History (2/23/2004).

[13] www.acm.org/serving/se/min990128 (4/11/2004).

[14] "Performance norms" seem to be another way to talk about "standards of professional practice".

[15] David Lorge Parnas, "On Code Reuse", *Software Engineering Notes* 24 (May 1999): 4

[16] David Lorge Parnas, "SDI: A Violation of Professional Responsibility", *Abacus* 4 (—— 1987): 46-52. Reprinted in (among other places): Deborah G. Johnson, editor, *Ethical Issues in Engineering* (Prentice Hall: Englewood Cliffs, New Jersey, 1991). Quote in Johnson, p. 15.

[17] Don Gotterbarn, Keith Miller, and Simon Rogerson, "On Code Reuse: a Response", *Software Engineering Notes* 24 (May 1999): 4-6.

[18] Gotterbarn, Miller, and Rogerson, p. 5.

[19] The others were: Fran Allen (co-chair), Barry Boehm, Fred Brooks, Jim Browne, Dave Farber, Sue Graham, Jim Gray, Paula Hawthorn (co-chair), Ken Kennedy, Nancy Leveson, Dave Negal, Peter Neumann, and Bill Wulf. *Software Engineering Notes* 24 (May 1999): 6. This is a committee of notables, but of academic notables [explain].

[20] For other examples of Gotterbarn's printed dissemination of the Code in 1999, see: Don Gotterbarn, "Two Approaches to Computer Ethics", *SIGCSE Bulletin* 31 (June 1999): 11-12; Don Gotterbarn, Keith Miller, and Simon Rogerson, "Software Engineering Code of Ethics Approved", *Communications of the ACM* 42 (October 1999): 102-107; and Don Gotterbarn, Keith Miller, and Simon Rogerson, "Computer Society and ACM Approves Software Engineering Code of Ethics", *Computer* 32 (October 1999):84-88.

[21] Gotterbarn\Version 4 IEEE Vote\SEPEC after 5.1\Dell\Shari1.

[22] Gotterbarn\Version 4 IEEE Vote\SEPEC after 5.1\Copyrigh. [sic]

[23] Gotterbarn once remarked in conversation that he now regrets failing to make notification a condition of use. With notification, he would have a better idea of how widely used the code is.

[24] www.acm.org/serving/se/Sepec (3/23/2004). Another version of this plan, indistinguishable except for its title date of "December, 1999", is on the IEEE-CS website: www.computer.org/tab/SWEC9912 (4/11/2004).

[25] www.acm.org/serving/se/min990414 (4/11/2004). SWEEP's plan, adopted the same day, has a fifteen month schedule (but no "termination date"). www.acm.org/serving/se/sweep (4/11/2004).

[26] Barbara Simons, Fran Allen, and Paula Hawthorn, "Not Now, Not Like This", *Communications of the ACM* (February 2000): 29-30.

[27] The inclusion of Hawthorn among the co-authors provided some balance. While Allen, along with ten other members of the panel, had opposed licensing, Hawthorn, along with two others (Browne and Parnas), had favored it.

[28] "ACM Panel on Professional Licensing in Software Engineering Report to Council (May 15, 1999)". www.acm.org/serving/se_policy/report (4/22/2004).

[29] For more details, Karen Hawthorne, "Engineering program accreditation to proceed at Memorial University", *Engineering Dimensions* (May/June 1999): 10.

[30] In addition, CCPE halted the process of accrediting Memorial's engineering programs—until the Supreme Court of New Foundland told CCPE it had gone too far. It could not hold up the accreditation of engineering programs as a way to pressure Memorial to reverse its decision concerning a computer science program.

[31] This meeting was held in conjunction with the 13th annual Conference on Software Engineering Education and Training, March 6-8, 2000, also in Austin.

[32] www.computer.org/tab/present/min20000304 (4/18/2004).

[33] The Texas definition was quite detailed, but (by allowing for "computer sciences") did not obviously exclude most ACM members likely to be interested in licensing: "The practice of software engineering will mean a service or creative work such as analysis, design, or implementation of software systems, the adequate performance of which requires appropriate education, training or experience. Such education, training or experience shall include an acceptable combination of: computer sciences such as computer organization, algorithm analysis and design, data structures, concepts of programming languages, operating systems, and computer architecture; software design and architecture; discrete mathematics; embedded and real-time systems; or other engineering education. Such creative work will demonstrate the application of mathematical, engineering, physical or computer sciences to activities such as real-time and embedded systems; information or financial systems, user interfaces, and networks." www.acm.org/serving/se_policy/report (4/22/2004).

[34] ACM members were not notified of the decision until a "Summary" was published on the ACM website (www.acm.org/serving/se_policy/selep_main) on July 7, 2000 (and made final on July 17). Don Bagert, "ACM Withdraws from SWEcc", FASE (*Forum for Advancing Software engineering Education*) July 2000. www.cs.ttu.edu/fase/v10n07 (4/18/2004), p.11.

[35] www.acm.org/serving/se_policy/selep_main (April 22, 2004). The authors of the report on licensing were: John Knight, Nancy Leveson, Michael DeWalt, Lynn Elliot, Cem Kaner, and Helen Nissenbaum. The final version of this report (dated August, 2001) seems to have been published (or, at least, "last modified") on October 12, 2001, bearing the copyright "@2001 John Knight and Nancy Leveson, all rights reserved".

[36] The third author, Michael Gorlick, had worked at Aerospace Corporation for more than twenty years on such projects as "wearable computing, grid computing, peer-to-peer infrastructure, large-scale data mining for structural genomics, internet appliances, and wireless devices. Sunset.usc.edu/gsaw/bios/gorlick (April 23, 2004).

[37] The ACM did not make clear in what sense software engineering either is or might be a "profession". Insofar as software engineering had a code of ethics of its own (and seemed to be trying to live by it), it was by this point a profession separate from engineering—whether or not it was a "computing discipline" or an "IT profession".

[38] FASE, July 1999. www.cs.ttu.edu/fase/v9n07 (4/18/2004).

[39] FASE, February 2000. www.cs.ttu.edu/fase/v10n02 (4/18/2004). This "Institute" is not to be confused with the Software Engineering Institute. According to its website (http://www.isri.cs.cmu.edu), "[i]n addition to participating in the School of Computer Science's top-rated undergraduate program in Computer Science, ISRI offers: A PhD Program in *Software Engineering*, A PhD Program in *Computation, Organizations and Society*, Professional Masters programs in *Software Engineering* and *Electronic Commerce* Distance education in *Software Engineering* with and without academic credit." Italics in original.

[40] Indeed, the problem for computer science might be even worse in some states, those that require anyone teaching "engineering design" to be a licensed engineer.

[41] FASE, July 15, 2000. www.cs.ttu.edu/fase/v10n07 (4/23/2004), p. 14.

[42] Email (Gotterbarn to Davis), March 14, 2001.

[43] http://www.computer.org/certification/cert_for_you.htm (August 6, 2004).

[44] John Knight et al, "On Licensing of Software Engineering as a Profession: Final Report of an ACM Task Force, August, 2001", http://www.acm.org/serving/se.policy.html (February 8, 2003). While the report also endorsed curriculum improvement as a way to improve software, it expressed considerable doubt that the Software Engineering Body of Knowledge (SWEBOK) would become a useful document any time soon. Apparently, SWEBOK was still linked in many

ACM minds with licensing. See, for example, J. Barrie Thompson, "Any Real Progress, or Is it Just Politics and Turf Wars?" FASE (September 15, 2001), p. 18. www.cs.ttu.edu/fase/v11n09 (5/21/2004), p. 14 : "The real difficulties are obviously associated with the Texas model for licensing software engineers and it is really unfortunate that the SWEBOK project is viewed by many (perhaps wrongly) as being inextricably linked to the Texas actions."

[45] "Professional Practice Committee Meeting 11/7/2002". For explanation of this report, see Gotterbarn's email to me (April 28, 2004): "IEEE revises technical documents—the IEEE professional practices committee asked me to report to them in 2002 on the status of the Code, I have attached draft of that report". The report is an attachment to this April 28 email. The Committee can only be found on the IEEE-CS website at http://www.computer.org/csinfo/standingcoms.pdf (September 20, 2004). The Committee, though apparently predating SWECC, seems to be acting as its successor (within IEEE-CS, of course). Its agenda for November 7 certainly resembled SWECC's. There is, for example "liaison" with the "SWE Curriculum project", the "SWEBOK Guide (Ironman Version)", and "SWE licensure activities" (among others). The Committee had ten members (beside Tripp). One of them was Dennis Frailey. FW PPC #42 7 NOV 2002.

[46] The report lists the following website as its source for AITP adoption: http://www.aitp.org/downloads/AITPAdoptsthe SoftwareCode.doc. When I checked that site on August 11, 2004, it contained no reference to the SE Code.

[47] On April 28, 2004 (email to Davis), Gotterbarn reported a German and an Arabic translations "in progress". All "official" translations are posted on the website of Gotterbarn's Software Engineering Ethics Research Institute: http://seeri.etsu.edu/Codes/default.shtm (August 15, 2004). All seven of the translations mentioned in the 2002 report are there.

[48] On July 18, 2004 (email to Davis), Gotterbarn announced in a postscript: "Did I tell you that the Australian Computer Society and the Australian Institute of Engineers have adopted the Code[?] That is especially interesting because it gives it a 'semi'-legal status. Following it is part of your professional obligation. I have been having discussions with the New Zealand Society and they are likely to adopt in a few months."

[49] *Forum for Advancing Software Engineering Education* 11 (September 15, 2001), p. 3. http://www.cs.ttu.edu/fase/v11n09.txt. By then SWEBOK had been examined by almost "500 reviewers in 40 countries" (during one of three reviews). Bourque email, May 29, 2003.

[50] http://www.swebok.org/home.html (August 11, 2004).

# Epilogue: Lessons for Code Writers, Theorists, and Researchers[1]

"History is philosophy from examples."
—Dionysius of Halicarnassus, *Ars Rhetorica,* xi. 2

## 13.1 What we can learn from this history

It is often said that "those who do not learn from history are doomed to repeat it." Why "doomed"? If history consists of noble achievements, of getting things right, we would be happy to repeat it. There would be no "doomed" about it. What ignorance of history dooms us to repeat is, it seems, the past's mistakes. History (not the past itself but our rendition of it) consists largely of mistakes from which we are supposed to take the lesson, "Don't do that!" Yet, according to one of the great students of history, "What experience and history teach is this—that people and governments never have learned anything from history, or acted on principles deduced from it."[2] The lesson of history is that we do not learn its lessons; we are all doomed. Though sweeping enough to be widely quoted, this lesson seems too bleak to be true.

I do not deny that history, including this history, consists, in large part, of mistakes. How could it be otherwise? Humans make a great many mistakes and find those of others particularly interesting. That is why Hegel thought the history of a happy people (one for whom things generally go well) a "blank".[3] Yet, it is success, not failure, that we want for our own projects; the successful, not the failed, that we admire and want to copy. The past, even when it is happy, seems to tell us something about the present, about how to shape the future, something we want to know because it could be useful, telling us what to do and what not to do. And, in some fields, such as engineering, we seem to have evidence that the past has been useful in just this way. Keeping good records is central to engineering. Engineers study their mistakes, develop routines designed to avoid making those mistakes again, keep to those routines as long as they seem to work, and in fact are much more likely to make new mistakes than to repeat old ones. Engineering certainly seems to testify to our ability to learn from history—or, at least, from the past.[4]

Even so, learning from history is not without paradox. Marx once observed that while great events may occur more than once, the first time is tragedy while the second is farce.[5] The difference between tragedy and farce is dramatic enough to suggest that Marx doubted that history could repeat itself. The second instance always differs in at least one way from the first. While the first is novel; the second is not. The existence of a predecessor is part of what converts the "tragedy" (the inevitability) of the first into the "farce" (the unnecessary and therefore laughable imitation) of the second. Yet, if history does not repeat itself, how is it possible to learn anything useful from it? Why is it that an engineer who makes an old (and important) mistake is generally out of a job (fired for "incompetence") when a new mistake would have been excused as "just one of those things"? What can we take from one circumstance and apply in circumstances that always differ in many ways?

For practical people, the answer is obvious: we learn from history just as we learn both from today's experience and even from the imagined experience of fiction. We simply "see" the similarities in different circumstances, adjusting our conduct in new circumstances to take account of what is old in them. What is not obvious is how we do that. Indeed, explaining how

such “seeing” is possible is the domain of both traditional epistemology and contemporary philosophy of science—and questions belonging to any domain of philosophy are questions having at least two plausible but inconsistent answers (and, often, several more than two). I can, therefore, make only a modest claim for the “lessons” I will draw from this history.

The lessons of history seem to be no more than rules of thumb, mere presumptive approximations of what will stand up to experience we have not yet had. If following a certain rule of thumb seems unlikely to turn out too badly, we follow it. If following it ends up badly more than rarely, we develop “doubts” but stick to the rule until we find a better one (since even a rather unreliable rule is better than none). There is no rigor in either the presumption or the approximation. We cannot rerun events to determine what actually caused what; we simply guess and go on. While we can test our guesses against experience, our experience seldom, if ever, speaks decisively. Today’s apparent success (or failure) may turn into its opposite tomorrow. Though every history has a last page, the past does not—or, at least, by the time it does, it will not matter. We are always like the man who, jumping from a tall building and being asked part way down, “How goes it?” responds, “Very well—so far.” When should we declare the “experiment” over? The end of the world is too late—and too hypothetical.

The “lessons of history” are interpretations of events, interpretations that (as T.S. Elliot put it) “a moment may reverse.” These lessons cannot be about the future. The future will resemble the past only roughly. (Almost every day has its surprises, some nasty.) The lessons of history must be about something we can know—without knowing the future—possibilities, what *can* happen (rather than *will* happen). Whatever else history can show, it certainly can show what is possible, possible not simply in the abstract logical sense that unicorns are possible but in the practical sense in which (as we have learned) even an “unsinkable ship” can sink (and that therefore even an “unsinkable ship” should have enough lifeboats for all passengers and crew). Lessons about what is possible in practice are valuable as guides because they wake us to considerations that might doom us if we miss them. No substitute for judgment, these lessons merely aid it, much as a “check list” does. The lessons of history alert us to ways in which we might improve our deliberations. They do not prophesy. Even when stated as imperatives, the lessons of history should be treated as advice: “Think about this.” That, anyhow, is the spirit in which I offer this Epilogue’s “lessons”. They neatly divide into three categories: lessons for those writing a code of professional ethics (13.2); lessons for theory (13.3); and lessons for those interested in undertaking research much like this (13.4).

## 13.2 Some Lessons for Code Writers

The events recounted in the preceding chapters suggest at least the following lessons:

13.2.01. **Keep the writing of a code of ethics separate from any discussion of licensing (or other controversial projects).** Though some models of profession treat licensing as a defining feature, in practice licensing raises many administrative questions merely having a code of ethics does not. Hence, keeping the question of having a code of ethics separate from questions of licensing, even from the question of whether following the code should itself be a condition of keeping a license, should reduce the number of obstacles in the way of adopting the code. The chances of a would-be profession adopting a code of ethics is, at any time, sufficiently low that not adding to the obstacles seems prudent. (Had the Joint Steering Committee taken even the limited interest in licensing that SWECC did, it might have had as short a life as

SWECC, and the Code might have aborted.) There is also a theoretical reason to keep writing a code separate from licensing. The claim that licensing is a defining feature of profession is itself controversial.[6] That controversy may have practical implications. At least some practitioners take theories seriously enough to vote against a code that seems to them to violate their theory. A code destined for use in licensing may have to allow some conduct a code designed for individual guidance need not. Much of the difference between engineering's two major codes seems to have this origin. The Code of Ethics of the National Society of Professional Engineers was designed to be used by state boards of licensure; the ABET code was not.

13.2.02. **Keep the drafting committee small**. Preparing a first draft of a code is not an activity made lighter "by many hands". It is more like the soup that "too many cooks" spoil. That is why the IEEE's Standard writing procedure, followed in Melford's Guide 4.1.4, advises:

> The WG Chairperson should then identify an individual to author an initial draft. The author should be permitted to prepare the draft with or without additional input or assistance from any other WG members, at his or her discretion.[7]

Every code of ethics needs what Mechler called "a Thomas Jefferson" to do a first draft (5.2). The drafter has two problems that need to be solved before large numbers of people become involved in the drafting. One is content, determining what should be in the first draft. The other is criticism. Even a "first draft" should go through several stages before being declared a fit subject for public discussion. The entire committee concerned with drafting the code should not exceed ten; its core should be no more than three or four. The actual drafting should be one person's work (though there might, for example, be one drafter at an early stage and another later); the early criticism, the work of half dozen or so.

13.2.03. **Keep preparation for the first draft simple**. The problem of content can be solved in several ways. The Gotterbarn-Mechler method is one. Have a small working group read as many other codes of ethics are possible, looking for provisions its members like. Make a list of provisions anyone likes. Circulate the list, asking whether any more provisions should be added (and what those provisions should be). Continue expanding the list until it seems relatively complete. (Provisions to which anyone strongly objects should be put to one side to be revisited once there is a rough draft.) When the list seems complete, give it to the drafter.

Another way to generate such a list is to hold sessions at the appropriate professional meetings. A member of the drafting committee presents some "ethics cases" to resolve—and invites the audience to suggest others that are "different". The audience then seeks to resolve as many of the cases as time allows. Part of resolving each ethics case is stating a "value" or "rule" supporting the resolution. Any value ("candor") or rule ("Don't mislead") the audience finds at all persuasive goes on a list from which the draft will derive content. The drafting begins when the list seems to have become stable (or when the meetings no longer surprise those who lead them).[8]

There are other ways to develop the *initial* content of a code. The exact way does not much matter because the *exact* content put into the first draft does not much matter. Mistakes of content can be fixed later. All that is important about the initial content of the code is that it be, by and large, appropriate to the profession in question and that there be enough of it to get the drafter started.

The history of the Software Engineering Code nonetheless shows that not all procedures are equal. Indeed, that history definitely recommends against two strategies. One is "divide and conquer", breaking work on the code into parts; the other is choosing the architecture before drafting begins, the "Scope" statement.(4.3) These two strategies share a common mistake. They assume that we know more about the future code than we can know without knowing what it will say. Dividing the code into parts assumes we know what the parts of the code will be. A comparison of SEEPP's original division of working groups and the Code's final list of eight Principles provides clear evidence of how hard it is to know even the major points to cover. The main problem with choosing the architecture before the content is not that some structures (say, a short code) may rule out a good deal of content (although it certainly may). The problem is that the architecture of a code should serve its content. Debating the architecture without knowing the content is likely to be uninvitingly abstract, the domain of a few monomaniacs. Gotterbarn's working group was saved from that abstract debate only because no one cared enough about the abstract architecture to respond to his call for discussion of the Scope. Mechler's SEEPP/E was saved in exactly the same accidental way.

13.2.04. **Get a draft as soon as possible *but do not circulate it to any authoritative body or individual until the draft is "final"***. Like all writing, code writing occurs largely on paper. Without a document, it is hard to do anything useful. To get a good draft may take years, but getting a "working draft" (that is, a draft good enough to work with) need not take more than a few months. Any committee (task force or working group) assigned to write a code of ethics should try to get a working draft as soon as possible. But, having got one, they should be slow to share it with the body that appointed them—or anyone else in a position of importance. We like to show off our accomplishments, and even a first draft feels like an accomplishment. We must therefore take care not to do "what comes naturally". Anderson's experience with the ACM Executive Council (3.3) and Mechler's experience with the Joint Steering Committee (5.6-5.7) tell the same story. Governing bodies are likely to have opinions about what a code of ethics should contain and to express them if asked. Having expressed those opinions, such a body is likely to find it hard to take them back. And having heard those opinions, the drafting committee is likely to want to write for the governing body rather than for the membership as a whole. That would not be such a bad thing if the drafting committee actually knew how the governing body would vote on particular provisions (or on the code as a whole). Generally, though, that knowledge is unavailable. In its place are the statements of a few outspoken members of the governing body or, at best, the body's first reaction, something not necessarily even close to how it would vote if it were voting on a final document.

13.2.05. **Have a well-defined procedure in place for turning the first draft into a final draft**. Those setting up a committee to draft a code should provide such a procedure. When Barbacci advised Gotterbarn and Melford to follow the IEEE Standards procedure, he did just that. However complex that process was, it turned out to be less complex than the one SEEPP fell into when it proceeded *ad hoc*. Because codes are never really finished, an "open ended process" must go on until arbitrarily cut off. Sometimes, especially in small groups (such as Mechler's), participants may understand the arbitrariness as necessary, especially if exercised late enough in the process that participants have begun to tire. In larger groups, however, there are likely to be some who resent the arbitrariness, becoming enemies of the code because they object to the process. So, if there is no set procedure in place when the drafting committee is appointed, one of the committee's first acts should be to define one, a procedure to follow from

first draft to final adoption. The drafting committee should make the rules governing its procedure public as early as possible to avoid unnecessary misunderstandings. It should try to get official approval of the procedure but not treat silence as disapproval (or even failure to approve). And it should stick to the procedure (so long as not disapproved)—unless it discovers it has made some serious mistake. Most of the remaining "lessons" concern this public procedure.

13.2.06. **Make the procedure as open as possible once there is a first draft.** The openness of the procedure has two (related) functions. First, it protects the code from the influence of prominent but eccentric individuals who would otherwise "represent" those less prominent. We have seen how often prominent individuals may misjudge the opinions of those they try to speak for. Second, the openness protects the drafting committee both from that error and the error of forgetting about practical considerations altogether (always a tendency in so high-minded an enterprise as drafting a code of ethics). Recall how many "aspirational" provisions entered Version 2 or Version 3 of the Code only to be ejected by Version 4. The IEEE's system of voting with comments is one way to open procedure, especially in a large organization or collection of organizations. Anderson's presenting of drafts at sessions of professional meetings, taking comments, and trying to revise in response before the next professional meeting, is another (3.3). Gotterbarn's survey is a third (8.1-8.2). Even Mark Kanko's survey of his students provided much useful information of this sort (6.7). What all good open procedures share is the drawing in of many people who, until then, had no connection with the drafting but are a fair sample of much of the code's actual audience.

The IEEE Standards procedure has at least three advantages over the others (above) worth noting (even though it was designed for technical standards, not a code of ethics). First, it assures a mix of new people in a way the others cannot. Anyone opposed to the code is likely to object that the code ignores such-and-such a constituency. The IEEE procedure (more or less) automatically answers that charge before it is made (at least for any substantial constituency). The procedure incorporates that constituency into the voting. Second, the IEEE procedure, though selective, is likely to bring in more participants than ostensibly more open procedures. The number of people participating in any way in the writing of a code of ethics is likely to be small. Gotterbarn's original world-wide call for participants brought in less than sixty names. His survey, published in magazines with a combined subscription of perhaps a hundred thousand, brought in just over sixty ballots. The IEEE procedures brought in an initial vote of just under a hundred fifty. Apparently, people are more likely to participate in response to a personal invitation than to a general call. Third, the IEEE procedure sets a limit on how many people need to be brought into the process (enough to have a good representation of each important group). The limit avoids the charge that too few people were involved, an important charge to avoid because the process from drafting to final approval seems unlikely to involve numbers large enough to refute the charge directly.

13.2.07. **Email is no substitute for in-person meetings.** Email seems to have been an important new tool for helping to open the process of writing the Code to people who might not otherwise have participated. They seem to have participated by email when they might not have participated by phone, fax, or conventional mail—in part, it seems, because email is so much easier to use. Email is, however, much more like phone or fax than like a meeting in person. Much of the work of writing the Code was accomplished through ordinary meetings of two to six people. This is striking given the original commitment to do as much of the work by email as

possible. Also striking is the importance software engineers themselves assigned to face-to-face meetings; even meeting by conference call was plainly "second best". This is striking because software engineers are just the people we might expect to be unduly partial to email (and all the other new technologies).

13.2.08. **Plan on a slow process from first draft to final adoption.** There are at least four reasons for preferring a slow process to a fast one. First, rushing tends to increase the number of errors, not only small errors (such as "typos" or unnecessary variety in language) but even big errors like dropping provisions that would otherwise have been adopted or including provisions with relatively little support. Second, a slow process tends to build support. People have time to consider arguments, to learn how many others agree with them, and even to gather information. A slow process allows for more debate, more revision, and even more "politicking". No one can (justifiably) complain that the code was "rushed through". Third, a code of ethics adopted too quickly is, all else equal, more vulnerable to repeal than one that has had to win adoption more slowly. Fourth, planning for a slow process forecloses another sort of complaint. *So long as the work is "on schedule"*, there are likely to be few complaints about lack of progress even if the committee takes half a decade to write a code. There will, however, be complaints much sooner than that, perhaps even within a year, if there is no schedule, just as there would be if work were plainly "behind schedule".

13.2.09. **Find ways to test the code by making people use it ("user testing").** This lesson, though related to the two preceding ones, makes a different point. The problem that testing the code is to solve is not that of getting the right content or wide participation, but of getting a code that is "user friendly" (one that suits those who are to use it). Claims that such-and-such is the "right way" to write a code are many. Some people claim that codes of ethics should be short, or they will not be used; others, that they must be long or they will not say much of use. Some claim that a code should include principles of interpretation or it will be misinterpreted; others that the interpretive principles will not be used and are therefore a waste of space. And so on. As far as I can tell, none of these claims about how to make professional codes user friendly rests on much more than anecdote or personal preference. Holding sessions at meetings at which members of the audience are given a copy of the code, asked to use it to resolve some cases, and then asked to comment on the ease of use, is one way to test a code. Comparing those responses with responses for a code with the same content but a different format would provide comparative evidence. Another way to do this, less expensive in time, is to use students in the appropriate professional program as the research subjects. Mark Kanko's students seem to have responded to the software engineering code much as did mature practitioners (6.7). If that is generally true, then professional students could be a convenient stand-in for practitioners.

What is actually needed is a systematic study of codes the results of which would be available in print to anyone about to write a professional code. Until we have such a study (or, better, several of them), the least anyone writing a code should do is some testing to identify serious impediments to using the code. Right now, the choice of code format seems to be one of those domains of expertise in which "nobody knows much" and "the louder they talk, the less they know". Anyone who claims to know anything about code format should immediately be asked for the evidence.[9]

13.2.10. **Work for consensus** (rather than simple majority). Ideally, a code of ethics should consist of those standards everyone in the group, at her rational best, wants everyone else

in the group to follow even if that means having to follow them to. Because people are seldom at their rational best, it is impractical to demand that everyone actually agree to the code. Yet, a bare majority, though practical in the sense of being relatively easy to get, is not a very strong indication of what members of the group at their rational best would agree to. Two-thirds, three-fours, or consensus (no strong objections) is a much better indicator. These higher levels of support, though harder to get than a bare majority, are practical in the sense of being useful (as well as possible). A code that survived by one vote today may die tomorrow with the change of a single one vote. A code that had three-fourths of the votes today is unlikely to face strong opposition tomorrow.

There are many ways to build consensus. The IEEE Standards process is just one. But it is a clever one, since it is designed to move from decisive support (two-thirds) to near unanimity in a way those participating are likely to respect. It commits people to reasons as well as votes; then responds to the reasons in ways likely to get the negatives to change their vote (not only as a technical requirement but with real conviction). Anyone designing a procedure for adopting a code should try for similar effects. The governing body should not find adopting the code a hard decision. The process by which the code reaches the governing body should vouch both for its workmanship and for its support among the general membership. As the IEEE's 1987 code showed, the life of a code of ethics lacking widespread support can be quite short. The process should build support.

13.2.11. **Get a writer to serve as the drafter or at least to work over the draft both at an early stage and again near the end.** By a "writer" I do not mean a grammarian, lawyer, or poet, but (merely) someone whose other work shows that she can write a clear sentence, order sentences so that one seems to follow from another, and produce large works that are a pleasure to read and easy to outline. Steve Unger is a good example of a writer in this sense; Keith Miller may be another. The writer need not have much to say about the substance of the code. Indeed, it may be an advantage to know no more than enough to ask, "What does this mean?" when a sentence is in fact unclear. Version 5.2, though improved in many ways since Version 1, clearly suffers from not having a writer work it over before adoption. A writer would, for example, probably have revised Principle 1 ("act consistently with the public interest") so that it is parallel to Principle 2 ("act in a manner that is…consistent with the public interest").

13.2.12. **Keep objectives modest.** I have already noted how small a group is likely to constitute the core of the writing process, and how few can be expected to take part even in a process of ratification as open as that Gotterbarn presided over. Hoping to involve a few hundred in the process of drafting, revising, and adopting a code of ethics may be reasonable; hoping to involve thousands probably is not (or Gotterbarn surely would have recruited many more participants than he did). Much the same is true about other aspects of writing the code. The objective of writing a code is to get a code that is "good enough", that is, a code (almost) everyone can live with, learn from, and have ideas about how to revise. Over all, it is reasonable to try to make small improvements on the procedures by which earlier codes were written, on the code's form, and on its content. To try for "perfection" in procedure, form, or content is not. Anyone who demands perfection will get nothing.

13.2.13. **Do not list "authors", "contributors", or the like in the code itself.** The Software Engineering Code followed the ACM's in listing names of individuals at its end, indicating that they were members of the body that developed the code.[10] In both cases, the list became a permanent part of the code. This was, I think, an innovation. I know of no other code

that has such a list. Credit for helping to prepare a code is usually given in an accompanying report or in the minutes of the appropriate meeting. The names are soon forgotten. To give credit in the code itself may seem a good way to repay the contributions of volunteers who receive no other reward for their work (except the satisfaction of having done it). But there are at least two objections to the practice. One is that including a list of names in the document adds to the cost of reproducing the document, and to what one has to get through to use it, without adding anything to its utility. A code of ethics is not an academic publication but a tool to be used in making difficult decisions. It should be designed accordingly. The second objection is that the list is likely to be misleading. Those listed will not all be listed for the same reason; their contributions may be quite different—and, indeed, some may have made no contribution at all, except signing up for the task force. When I first asked Gotterbarn about his list, I described it as the list of "signers". He objected:

> At each go round of the Code, different people had differences of opinion. We all worked on a document and I don't think anyone, including myself[,] is completely satisfied[,] so I guess it is wrong to say the people on the list "signed off" on the Code.[11]

Gotterbarn also did not want me to describe the list as consisting of those who "contributed" to the Code because there "are at least two names on the list that never contributed anything." Why were those names still on the list? Gotterbarn could not, he thought, simply remove them once he had put them on—even if he put the name on only because the person named had once signed up for the task force:

> There is no way to remove a name from a volunteer task force, so I subtly sent out a message asking people to say how the[y] wanted [their] name to appear on the task force list. Yes[, they] did send a preferred form for their names. So the [name] appeared on all of the publications as members of the task force. Notice the word "active" or "participating" does not appear in front of the word "member" on the task force list.[12]

Gotterbarn may not have been able to avoid the pettifogging distinction between "participating" and "non-participating" members of the task force. But he could have avoided having that problem become a permanent attribute of the Code. He need only have followed standard practice, omitting all names of individuals from the Code itself.

13.2.14. **Never suppose that there are experts on what a code should say**. Over the six years during which the Software Engineering Code developed, a fair number of people, important and not so important, made claims about what a code of ethics should or (more often) should not say. Many of these claims were, at one time or another, significant impediments to writing the Code. In retrospect, most seem pompous, theory-driven prejudices, with no basis in the common sense of software engineers or in any statistical study of what codes of professional ethics actually contain. There are, or at least may be, experts on what codes *in fact* say; and such experts could be quite useful in developing the initial list of provisions and later thinking about how that list might be augmented.[13] But any code of professional ethics is simply a set of (morally-permissible) standards to guide members of the profession in ways they (at their rational best) want each other to conduct themselves. We have no procedure by which to learn what members of a group will want at their rational best except by asking them what they do

want, challenging those answers with reasons, and seeing whether their preferences change. Their decisions, after due deliberation, are the best indicator of what the standards should be (though even that indicator is fallible). An expert may be able to spot contradictions, infelicities in expression, or provisions that have a history of causing trouble. What experts cannot identify is provisions that must be included in any code of ethics or provisions that (though morally permissible and competently written) do not belong in a code of ethics.[14]

13.2.15. **Don't expect "blue ribbon" committees to do much work**. A "blue ribbon" committee, that is, a committee of those highly respected in a profession, may be useful for some purposes, such as protecting a proposed code of ethics from criticism. What blue-ribbon committees seem unable to do is write a code of ethics (or other document). There are several reasons for this. Three seem to be particularly important. First, most members of a blue-ribbon panel are likely to be too busy to write much. The code will be just one of many projects, one without (much) institutional support. Second, the skills of the highly respected in a profession generally are elsewhere, no profession having much opportunity to earn a reputation writing codes of ethics. Third, a blue-ribbon committee is likely to consist almost entirely of people successful enough to have hardened in their views and far from the work most members of the profession now do. Codes seem to be written instead by the "middling sort" of professional, long enough in the field to know what it is like but not so successful to be insulated from its pedestrian concerns; they are also likely to be disproportionate academics, even in fields where academics are not otherwise prominent. Many of these academics may (like the ACM's Anderson, 3.3, or me) come from outside the profession itself.

13.2.16. **Start planning early for dissemination, education, and administration.** A code of ethics is simply a piece of paper unless it enters practice. The first step to helping it enter practice is publication. "Publication" used to mean printing—in books, journals, or stand-alone pamphlets. It still does—in part. But web publication has also become an important means of dissemination. An organization that lacks its own web page may still place its code on other web sites. Among these others is IIT's Codes of Ethics Online (ethics.iit.edu/codes).

Codes are, however, not necessarily used just because they are published. They can be daunting documents and, in any case, they are not self-interpreting. Practitioners need help with interpretation. Classroom instruction is one way to provide that help, whether in an undergraduate class or in a session at a professional meeting. Supporting documents, guides to the use of the code, are another way to help members of the profession use the code. Professional societies can also establish "ethics committees" to receive and resolve inquiries from members concerning the interpretation of the code. These resolutions ("opinions") may be formalized and published in the society's journal, in a collection (such as those of the National Society of Professional Engineers, the American Medical Association, or the American Bar Association), or in some other way.

Providing authoritative interpretations of the code is part of its "administration" (as well as part of its dissemination). A professional society may also arrange for adjudication of disputes among members concerning "unethical conduct", for investigation of complaints against members concerning violations of the code, and for other quasi-legal procedures. Like licensing, however, quasi-legal enforcement is likely to generate opposition to the code, even if the process of enforcement is largely or entirely educational. Few people like to be told something they have done is unethical—especially if they are told so by an organization the judgments of which their colleagues or employers are likely to hear of and take seriously.

13.2.17. **Establish a procedure for review and revision.** Like people, codes age. What was up-to-date in 1970 may sound ancient even thirty years later. Part of keeping a code a part of practice is providing for regular reexamination. And even if codes did not age, they would be imperfect, with experience revealing unexpected imperfections or confirming the existence of imperfections already suspected. While it is never possible to have a perfect code, it is always possible not only to improve what age has damaged but to improve what has aged well. The opinions of an "ethics committee" will, from time to time, identify provisions in need of rewriting and even, now and then, a provision that everyone can see should be there but for some reason is not. Where there is no ethics committee, the work of looking after the code can be given to a body with wider responsibilities, such as SWECC was, or to a temporary committee established, say, every five or ten years (for example, in every year ending with a zero). Such a body could carefully review the entire code, survey the users, or compare it with other codes, propose revisions based on what it learned, and then dissolve. Those who write a code of ethics should, if possible, send their governing body, along with the final code, a recommendation for a permanent or regularly reoccurring revision committee. That recommendation should include a procedure for moving from proposals for code's revision to final approval. The procedure should be, more or less, the same as the original procedure for adoption of the code.[15]

13.2.18. **Be wary of official translations of code.** Translating an international code of ethics into the native language of those likely to use it may seem like one of those ideas to which no one should object—and perhaps it is. But it also has consequences worth taking into account before proceeding. A translation is, of course, never quite equivalent to the original. The same problems of choosing just the right formulation of a certain rule reappears all over again each time the code is translated into another language. Even software engineers fluent in both languages may disagree about whether the translation is close enough to the original. Each translation is a major commitment for those who undertake it. As the translations accumulate, revising the original code becomes a much more complicated process than it would be if the code were in only its original language. The translations should be tracked. Any amendment of the original should mean that within a few weeks or months all translations or, at least, all those posted on the internet, should be revised as well. Translation creates a problem of coordination, one that grows as the number of translations grows.

## 13.3 Lessons for theory

The story told here should teach us something about professions. But what it can teach us depends in part on what it is. I began supposing it to be a work of sociology. I was to describe, analyze, and (ultimately) understand a certain social process, the writing of a code of professional ethics. I have, however, come to think of the book more as history (because, as I reported in 1.5, I have come to trust the documents much more than the interviews). I can only offer a plausible narrative founded on the documents that events left behind (and interviews conducted well after those events). Even when "I was there" (as minor participant), I (the narrator) was not. My own memories granted no privilege even when I was trying to understand my own acts. However dirty the past, historians write with clean hands.

What can history teach us about a social process when all we get out of a narrative is what we put in? That is one of those "philosophical questions" that give philosophy a bad name. Of course we will get out of a narrative only what we put in, but what-we-put-in is not the same

as what we *consciously* put it. So, for example, many of the lessons of 13.2 (concerning how to write a code of ethics) were new to me. I had not thought of them before I reached the end of my story and could see it (more or less) whole for the first time. I put those lessons into the narrative without realizing it. The lessons were as much a surprise to me as some of Euclid's theorems were when I first encountered them in high school. Simple axioms (and postulates), *when put together in the right way*, justify conclusions we do not anticipate when accepting the axioms. Even Euclid would doubtless have found some of the theorems his successors derived from his axioms surprising. We seldom know all that our acts imply.

There is nonetheless a question about what my narrative can tell us of "the social process" of writing a code (or organizing a profession). The problem now is "social process" (not, as in 13.1, "history"). "Social process" is an abstraction, a product of social *theory*, something to be stated in general terms. What I have narrated are particular events. Though open to generalization, one narrative does not itself justify any (interesting) generalization. The justification of generalizations must await other similar studies. The lessons about social process we are to learn from this research are (in large part at least) a function of how research not yet imagined turns out. All one narrative can do is raise questions. Many of the questions take the form of a presumptive counter-example to a theory: such-and-such a theory says that all X's are Y's, but here is an X that is not a Y. Whether this presumptive counter-example actually is a counter-example will depend on whether the generalization it threatens can be revised to turn the example into no more than an interesting exception (or whether, upon re-examination of the evidence, the presumption simply disappears). A single narrative is, at best, an invitation to further research and theory.

Does my narrative offer sociology any such invitations? I believe it does. It certainly suggests that we know less about how, why, and even when professions write codes of ethics than commonly supposed. Consider, for example, Larson's classic characterization of (nineteenth century) professions:

> The professional sector of the middle classes aspired to gain at least as much prestige as the most acceptable commoners (or, in America, the most successful businessmen) within this social stratum. This social ambition colored the professional project; it is likely to have inspired the ambition of most of the professionals who responded to the calls of the early organizers, for it is obvious that efforts to secure a relatively new market would have resulted in some benefit for the self-seeking individual.[16]

If we treat this passage as a claim about all professions, including software engineering, we have reason to doubt that gaining the "prestige" of the "most successful businessmen" has much to do with the "professional project" (even "in America"). There is, first, the problem that none of the participants in writing the Software Engineering Code of Ethics ever expressed any interest in gaining such prestige, much less the expectation of gaining it through organizing software engineering as a profession. Indeed, the disparaging comments regularly directed at Bill Gates (and Microsoft) are evidence that whatever "prestige" software engineers want has little to do with success in business. The "prestige" software engineers explicitly said they were seeking—if "prestige" rather than "deserved reputation" is the right word here—is the "prestige" that comes from belonging to a group known to do software engineering well. Those organizing software engineering as a profession (apparently) wanted to avoid the embarrassment of working in an

occupation that produced poorly constructed software behind schedule and over budget. They did not want to be associated with what they considered incompetence.

Our research would not raise doubts about what the ambitions of those organizing the first professions were *if Larson's claims about them rested on the sort of historical evidence that our does.* But Larson has no such evidence. Behind her claims for what is "likely" or "obvious" is nothing more than the theory of economic rationality (that people act only, or at least largely, to serve their own private "self-serving" interests). Economic rationality explains much market behavior well. It does not, however, explain all of it. It does not, for example, explain why self-interested individuals would buy life insurance, (anonymously) contribute to charity, support their aging parents, or engage in any number of other acts important in most economies but at least apparently "altruistic". Most thoughtful people, including most economists, understand that a (purely) "economic man" would be a monster to whom the term "rationality" would apply only in a sense (the economic sense). The professions, or at least the writing of professional codes, *may* exist (in large part at least) outside the domain of economic explanation (just as most other forms of altruism do). That, at least, is a reasonable inference from the story told here.

But (it might be said) I have misinterpreted Larson. She did not claim that the "project" of self-interest was conscious; she merely spoke of it "coloring" the (conscious) project of establishing a profession. Indeed, she explicitly noted that sociological theory does not understand "project" in the ordinary sense (as a "planned undertaking"):

> it does not mean that the goals and strategies pursued by a given group are entirely clear or deliberate for all the members, nor even for the most determined and articulate among them. Applied to the historical results of a given course of action, the term "project" emphasizes the coherence and consistence that can be discovered ex post facto in a variety of apparently unconnected acts.[17]

In other words, "project" is probably the wrong word for what Larson has in mind. "Tendency" would be much better for two reasons. First, "tendency" does not suggest planning or even consciousness of the outcome in question. Second, "tendency" allows for degrees (just as "color" does). Tendencies can be strong or weak (while projects just are or are not).

Because a tendency can exist in any number of degrees, disproving its existence is much harder than disproving the existence of a project. For example, suppose (as I have claimed) that this book shows that writing the software engineering code of ethics was primarily (at least at the conscious level) an altruistic undertaking. Larson might agree and yet claim that she can show (using apparently unconnected acts) that it is "likely" that self-interest "colored" the project to *some* degree. Who can doubt that? The interesting question is: To what degree? And the story told here suggests that self-interest can offer little insight into the acts recorded here (or at least, little beyond what altruism already provides). Now, there may be another way to tell this story, one respecting the evidence at least as much as this one does but leading to (something like) Larson's conclusion. That cannot be ruled out *a priori*. But neither can it be assumed *a priori*. We are certainly not entitled to draw conclusions from such a possibility until it exists in a form that can be read, checked, and appraised.

I have picked on Larson only to make a point I could have made against many sociologists writing on professions, including many of the most important. The general tendency of all the social sciences is to look for explanations beyond subjective purposes. Sometimes this

search ends in an important insight (such as "the invisible hand" of economics). But sometimes it does not. Subjective purposes explain some events as well as they can be explained. The social sciences then have nothing (interesting) to add; there is only history.

I believe the best explanation of the events narrated here are the conscious motives of the participants. But I might be wrong. From the beginning, I knew that the events I—as participant—knew of were not the whole story. One reason for interviewing participants was to fill in parts of the story that the documents did not reveal. I was concerned, for example, about the possibility of secret meetings at which "the real decisions" were made, rendering the process I had observed mere epiphenomena. There may have been such meetings, especially when the ACM withdrew from the Software Engineering Coordinating Committee (SWECC). But I have found no evidence of "midnight meetings in smoke-filled rooms" at which decisions were made for reasons that could not bear the light of day. No one seems to have thought, or at least suggested, that the motives of participants even at the meeting at which the ACM voted to withdraw from SWECC differed in important ways from those that appeared in official reports or at the meetings for which we have the minutes, much less that the decisions taken in closed meetings require explanation in terms of processes not wholly conscious. When I asked about "the politics" of this or that event, I heard about bad decisions, hasty decisions, ill-informed decisions, but nothing that required me to look for social dynamics bypassing what people knew, intended, and consciously did. The reasons given in public seem to have been much like the reasons given in private and seem to explain the acts of the individuals in question. There does not seem to be any reason to "go deeper".

Of course, my focus has been on the writing of a code of ethics. Perhaps writing a code of ethics is not important enough to be subject to the social dynamics of concern to serious social theory. I must admit that possibility. While I have argued that codes of ethics are central to professions (1.4), that centrality remains controversial. For most sociological theories, the code of ethics is of little or no significance. The body of knowledge, the curriculum, and licensing are much more important.[18] Because most sociologists may reject my claim for the importance of codes, it is worth recalling that I have not ignored those features of profession most sociologists consider central. My story includes much about the body of knowledge, the curriculum, and licensing. Yet, there was nothing in my story to suggest that the code of ethics was unimportant or that development of the body of knowledge or the curriculum proceeded in a fundamentally different way. The one surprise in my story is licensing, the primary means of achieving the market monopoly at which many sociologists assume professions aim. Licensing was so controversial *among* software engineers (or, at least, among computer scientists) that the ACM withdrew from SWECC just because SWECC *seemed* to provide support for licensing. (12.4) Licensing, though important, was important in exactly the way social theories like Larson's say it should *not* be.[19]

Whether history presents a good counter-example or not, it may raise new questions for research. For me (as a philosopher), the most interesting question that my narrative raises is why so many philosophers appear in this story of software engineering—and why they appear so prominently. Though lawyers outnumber philosophers ten to one in the population as a whole, philosophers outnumber lawyers in the writing of the Software Engineering Code of Ethics. This was true even though Gotterbarn sought out lawyers (and actually recruited one, Barber). Of the philosophers, two (Weil and I) were (more or less) a simple byproduct of this study. We would not have been there but for the research. Gotterbarn recruited two others (Fodor and Sullivan),

though not because they were philosophers. Gotterbarn also recruited one other, Fairweather, not because he was a philosopher, but as a member of the Centre for Computers and Social Responsibility (CCSR). Weckert volunteered, but as a computer scientist, not as a (former) philosopher. Philosophers were not part of Gotterbarn's plan (even though Gotterbarn himself had once been a philosopher). The philosophers just accumulated.[20] They also seem to have been present in disproportionate numbers for the writing of the ACM Code of Ethics.[21] Why? What do philosophers contribute that "mere" computer professionals cannot? Or is the explanation of their unexpected numbers not what they contribute but what they find attractive? Does the pattern reappear in other professions?

There is a similar question about the role of academics. They seem to have a much larger role in the writing of the Software Engineering Code (and, indeed, in the writing of the ACM Code) than their absolute number would lead us to expect. Even in Mechler's little group, they accounted for more than half. (Of the seven, only Mechler, Norman, and Sullivan were not academics at the time they worked on the Code—and Sullivan was a former academic while Norman worked on the staff of a university and even taught a course most semesters.) Gotterbarn's informal drafting committee at CCSR (and his executive committee as well) consisted entirely of academics (though each had considerable experience of practice). Even the original list of SEEPP volunteers (1993) seems to have had a disproportionate number of academics (17 out of 29). Nothing in the documents I have examined suggests a tension between academics and non-academics such as exists in some parts of engineering—perhaps because (as we saw) the boundary between practitioners and academics in software engineering is quite porous.[22] The academics do not seem at all interested in seizing power or using the code to feather their own nest. They seem to be there because they have more time to devote to the work—or, at least, better control over the time they have. But perhaps they are present in professional organizations in unexpectedly high numbers for other reasons too. What might those be?

## 13.4 Some lessons for researchers

Are there any practical lessons for those who might be considering similar research, that is, research into the writing of codes of professional ethics? I believe there are—though, as with the lessons for 13.2, these are best treated as "things to consider".

13.4.01. **For some purposes, a telephone interview is no substitute for an in-person (on-site) interview.** One review of our original NSF proposal suggested that we interview by telephone rather than in-person. The reviewer cited his (or her) experience for dismissing in-person interviews as an unnecessary expense. After reviewing the (quite small) literature on the question and consulting social scientists we knew who did open-ended interviews, we concluded that the reviewer must have been relying on experience with survey questionnaires rather than open-ended interviews. I now have four more reasons for that conclusion.

First, open-ended interviews tend to last much longer than survey interviews done over the phone. The typical phone survey seldom lasts half an hour. Our shortest interview—with Weckert—was just over an hour. I think there is a practical reason for the shorter duration of phone interviews. Twice during my research, I tried (something like) a phone "interview" (once with Douglas and once with Chikofsky). Each interview was at the suggestion of the interviewee after I contacted her or him about an in-person interview. In each case, the interviewee promised

to fill me in on "background politics" (as a supplement to the formal interview to occur later) and, in each case, the phone call lasted less than a half hour. The phone interview seemed to run out of energy in a way in-person interviews (generally) do not. Both times I was surprised at how tired I felt after the interview, especially how sore my arm and ear were. The interviewee likely felt the same. This, then, is anecdotal evidence that phone interviews may have to be shorter than in-person interviews in part at least because holding a phone to the ear is more tiring than ordinary listening. Perhaps too, there is another factor, the special attention required to listen to a speaker without the facial cues, gestures, and so on that one has when conducting an in-person interview.

Second, the in-person interview gives the interviewee a better opportunity than a phone interview to assess the interviewer. The interviewer has the same opportunity. The in-person interview can build a relationship in a way a phone interview cannot. A disembodied voice is not as easy to trust as a full human being. Several of my interviews ended with the interviewee and me together going to lunch or dinner. The interview was over, but we had discovered enough in common to continue the conversation. Most of the interviewees who later sent me documents, or answered questions I emailed, even several years after the interview, were the ones I shared a meal with. The in-person interview paid off in ways I would not have predicted.

Third, the relationship between interviewer and interviewee is different when they meet in person "on-site" (that is, at, or at least near, the home or office of the interviewee). There is, of course, the honor implicit in the time and money that the interviewer has spent to do the interview in person. A phone interview cannot do the same honor. The honor is much increased when the interview is convenient for the interviewee but far from the interviewer's home.[23] (Is the honor doubled if there are two interviewers rather than one?)

Fourth, I conducted about half the interviews in the interviewee's office (academic office, law office, laboratory office, or the like). I learned something about the environment in which the interviewee worked that I could not from a phone interview or even from an interview in my hotel room or at some neutral location. For example, when I entered the Software Engineering Institute (SEI) to interview Barbacci, the first thing I noticed (apart from the impressive scale of SEI's modernist building) was its security. Though located on a university campus, it was plainly not an (ordinary) academic building. There was just one unlocked entrance, with a guard just inside the door, a large desk before him, and a book on top of it into which visitors had to enter their name, "company", time of arrival, and host. After I entered that information, the guard phoned Barbacci, issued me and my graduate student each a "guest badge", and asked us to wait until Barbacci came to escort us upstairs. When Barbacci arrived, he too was wearing a badge, but his had his name and photo on it. The security procedures at SEI were even stricter than at IBM's Watson Research Center (perhaps reflecting SEI's relation to the Department of Defense or just its location on a big-city street). That introduction to SEI helped me appreciate Barbacci's informality (and candor) when we finally reached his office and began the interview.

13.4.02. **Think twice before doing in-person on-site interviews.** I have nonetheless come to sympathize with that anonymous reviewer who thought the in-person on-site interviews might not be worth the cost. Scheduling an interview took about as much time as the interview itself. Travel consumed much more time, especially after the changes in airport security that followed September 11, 2001. Anyone thinking about doing research involving interviews should think about whether in-person interviews are worth the time—as well as the expense—of travel to distant locations. While I think my interviews were worth the time and expense, some

interviews may not be, especially if the questionnaire is relatively short, the questions multiple choice or at least short answer, and the information to be solicited not very personal. Interviews at conferences also have attractions that interviews on-site do not, so long as there are several people to be interviewed there, or the site is close to the interviewer's home, or the interviewer would be there even if no interviews were scheduled.

13.4.03. **Take notes no matter what.** When we prepared our NSF proposal, we worried a good deal about whether to tape interviews or just take notes. Our social scientists advised us that taping would not significantly reduce the spontaneity of the interview. We (the philosophers) nonetheless had doubts. Now, having done (much the same) interview with and without taping, I agree with the social scientists. Our shortest interview, our first (Weckert), was not taped (because we did not think to bring a tape recorder); some interviews were not taped because the recorder failed to work. But no interview went untaped because the interviewee refused. Our interviewees, like most of those our social scientists had interviewed during long careers, did not seem to find the tape recorder intimidating. We did, however, have a few occasions when, in the middle of the interview, the interviewee asked to have the tape recorder turned off. In each case, the interviewee wanted to speak candidly about a particular individual, providing background for what was on the tape. In each case what was revealed was not a scandal but the interviewee's personal judgments, something (he or she thought) I—but not the world—should know. After the revelation, with perhaps some follow-up questions (and answers), I switched the tape recorder on again and continued the formal interview.

The problem with taping interviews was not that taping intimidated our interviewees but that it was never entirely convenient. There were always the awkward few minutes at the beginning of the interview when we tried to make conversation while seting up and testing the machine to make sure it would pick up the interview. That over, we could begin in earnest. These preliminaries were, however, not the only inconveniences of taping the interview. Tape for the portable recorders we used could not be made to last over 90 minutes while our interviews tended to last about 120 minutes. The tape would, then, have to be changed at least once (twice if we used a higher fidelity setting). If we heard the recorder click off, we would have to stop the interview to change the tape, breaking the interview's flow. Sometimes, however, the interview was so engrossing, or the click so quiet, that we missed it. Sometimes we soon noticed that the recorder had gone off, changed the tape, and switched the recorder back on. Sometimes we did not realize the tape had run out until the interview was over. (Once we even discovered that we had somehow failed to turn the recorder on.) We soon learned to treat the tape as auxiliary to our notes. Even when I interviewed alone, I took notes (though making notes slowed the interview process and broke the flow almost as much as, and much more often than, stopping to change tapes did).[24]

Another advantage of notes over tape is that turning notes into an "interview report" is much easier than turning tape into one. Transcribing a tape is slow. Few people can type as fast as people talk. Transcribing usually requires stopping, reversing, and replaying the tape once, twice, or even several times, every few sentences. Few academics have the patience for transcription. Giving the job to someone not present at the interview, even a commercial transcriber, has risks of its own, however. Transcribing is not necessarily reliable. Sometimes crucial words are not clear on the tape (even though they were clear to the interviewers). The transcriber should, of course, note any doubts in the transcription, allowing the interviewer to correct the text. But sometimes what the transcriber takes to be clearly one thing is in fact

something else. That will not be indicated in the transcript. The only way for an interviewer to catch it is for him to read the transcript while listening to the tape—and even then he may miss the change. If he does not catch it, the interviewee may—but only if the error is inconsistent with what she meant to say.

13.4.04. **What counts as "the interview" is not a question of fact.** Most of the interviews we conducted exist (or, at least, existed) in at least four versions (apart from the event itself). We have discussed two versions already. First, there are the notes one of the interviewers took and second (with a few exceptions) the tape of the interview. The notes (sometimes with assistance from the tape) were then turned into a formal document (the third thing that might count as "the interview"), our reconstruction of what happened. When I worked with a graduate student, the formal document was sometimes close to a transcription. When I worked alone, the interview document was really a summary (with a few good quotations), much shorter than a transcription of the interview would have been.[25] Once there was a formal document the interviewers agreed on, I sent it ("the edited interview") to the interviewee, who was to check it. Interviewees occasionally found that we misreported what they had said, but more often corrected their own grammatical errors or errors of fact. Occasionally, they toned down a phrase, substituted a formal expression for slang, or otherwise treated the document as a rough draft of their final remarks. What they sent back, we called "the approved interview".

According to the "protocol" (the explanation of the research that we gave to the interviewee along with the consent form), we were supposed to mark corrections interviewees made with "Interviewee's Comment". We interpreted the protocol to mean we should mark the corrections when, but only when, we disagreed with the interviewee about what was said at the interview *and* the correction was more than editorial. We never had such a disagreement. We interpreted the protocol this way because it allowed "additions" to be treated in exactly this way.

I naively supposed that the "approved interview" was "the interview" (and the earlier versions were merely stages in achieving that final document). My advisory board, or at least the two sociologists on it, disagreed. The approved interview was one thing, they noted, something careful and contrived, while the transcription (or tape) was something else, the (relatively) spontaneous expression of the moment. The factual and grammatical errors, the slang, the heated expressions, all told something about the interview itself (and so about the interviewee) lost in the revisions. While I conceded their points, I did not have the same concern about preserving the moment of the interview. The point of the interview, as I saw it, was to get the interviewee's help in reconstructing what happened. I preferred an interviewee's considered judgment to spontaneous expression.[26] I thought that the interviewee had a right to shape the interview to his or her satisfaction. For me, *the* interview was not the original oral event, but the final (approved) document.[27]

I was so sure of that that—for at time—I threw out my notes as soon as the interview was approved. Once I had the approved interview, the notes seemed to have no further use. Indeed, they might, in the wrong hands, threaten the confidentiality I had promised. I would have erased the interview tapes as well if I had thought of it. But I had initially bought such a large package that I threw the tapes into a box under my desk once I was done with them, thinking I might reuse them later (never supposing that within a few years digital recording would make tapes obsolete). The tapes still exist only because the advisory board, shocked when I told them what I considered "the interview" and what I intended to do with the tapes when I got around to it, emphatically advised me to save them.[28]

Having saved the tapes, our research group faced a problem not anticipated. Along with the consent form, each interviewee had received the research protocol explaining the research in greater detail. For our purposes (though perhaps not for the IRB's), the protocol was part of the consent form and as morally binding. What the protocol said about the tapes was:

> For our own research purposes, and for the benefit of future historians, we will tape record or take notes of this interview. We will transcribe or write up the notes and send you a copy for comment. You may suggest corrections or additions. Your corrections will be inserted at the appropriate point, but marked as "Interviewee's Comment". We will do our best to keep the text of your interview confidential—unless you notify us, in writing, that the interview may become part of our Public Archive.

There is nothing in the protocol about what will be done with the tapes of the interview (except for the reference to "the benefit of future historians"); and there is nothing at all about what will be done with the notes.[29] There is, however, a distinction drawn between "your whole interview" and a "summary":

> In addition, we plan to prepare a summary of your interview which—subject to your written permission (and also subject to the written permission of any living others named in the summary)—may be placed in our Public Archive as part of a database accessible on line. Should you give us written permission to place your whole interview in the Public Archive, this summary would also be placed in the on-line database. Nothing placed in our Public Archive is confidential. You may withdraw your material from it at any time up to the moment it is placed on line. Simply send a notice in writing to: [CSEP]

The advisory board interpreted the "whole interview" to be the tape—and the approved document, especially when short, to be the "summary".

While there is something in the advisory board's interpretation of the protocol worthy of Solomon, there is also the sort of problem that arises when someone uses Solomon's means (for example, threatening to cut a baby in half). The protocol says that "the portions of interviews for which you have requested confidentiality (including your subsequent comments) will be placed in a Research Archive available only to registered and approved researchers who will be bound by the same rules of confidentiality as the original project researchers." None of our interviewees has requested confidentiality for the tapes—in part no doubt because we gave them the impression that we were only going to use the tapes as aids to memory ("[for] our research purposes"). Should we keep the tapes confidential until January 1, 2020 (when "this Research Archive will have the original material removed and personal references restored to become a Historical Archive available for use by the general public and future historians")? Or should we keep the tapes confidential until the interviewee gives permission to reveal them (or until the interviewee dies)? If the interviewee signed a consent agreement allowing the "interview" to be put online, may we put the tape online? (When we wrote the protocol, putting tapes online was not technically feasible; now it is.)

For now, we are even asking consent to put "the summary" online. While no one has refused to give consent, about a third of our interviewees have still not responded one way or

another to our request. Should we put these summaries in level 1, 2, or 3? What about the approved (or merely edited) interviews? The Advisory Board is still considering these questions.

13.4.05. **Not all interviewers are equal; an interview teams including a philosopher may be better than one consisting entirely of social scientists** (anthropologists, sociologists, political scientists, historians, or the like)**.** I conducted about a third of the interviews with a graduate student in sociology, discussing the interviews with him afterward. I soon noticed that we did not pay attention to the same aspects of the interview. I recalled arguments, descriptions of career, and names of organizations; he recalled tone of voice, gestures, odd expressions, and the names of individuals mentioned during the interview (evidence of, he said, "networks"). During the interviews I sometimes departed from the questionnaire to learn more about how this or that job was done, the way one organization was related to another, or other details I thought my interviewee particularly qualified to help me understand. My graduate student would ask such questions as, "How do the people that are supposed to adhere to it see the code?" or "How much of this was driven by concern for the public and software engineer's relationship with the public and their safety?" He did not ask those questions because he believed the interviewee would know the answer. He was simply interested in seeing what sort of answer the interviewee would give. He was trying to understand the interviewee.[30]

The difference between our two interviewing styles strikes me as emblematic of the difference between philosophy and sociology (and so, between most philosophers and most sociologists). I do not claim that I was a better interviewer than he was—or even (what is more likely) that he was a better interviewer than I was. What I do claim is that we were somewhat different interviewers likely to uncover somewhat different information. Insofar as I am right about the difference and its relation to our different disciplines, having a philosopher interviewing along with a social scientist seems likely to improve the interview. Perhaps much social-science research would be better if philosophers were part of the research team not only helping to shape questions or interpret data but helping to collect the data too. Philosophers tend to remind social scientists of the (substantial) contribution of reason to human conduct; social scientists, of that part of human conduct reason cannot entirely explain.

13.4.06. **Think of research like this more as journalism than as ordinary social science.** The protocols, consent forms, and rules concerning confidentiality now governing social science research began in medical research, a response to a series of scandals. Medical researchers undertook to avoid repeating not only the crimes of Nazi medical research but also the deception of the Tuskegee experiments, quasi-voluntary research on prisoners, experimentation on patients and wards of the state without informing them, and other once-common but now clearly objectionable practices. Much social-science research resembles this sort of medical research insofar as the research subjects are young, unduly subject to pressure (students in a class, for example), poor (and so easily won over with small incentives), uneducated (and so easily misled), or otherwise especially vulnerable. Such vulnerable subjects of research in the social sciences need the protection of an IRB for much the same reason that they need that protection when subjects of medical research.

Our interviewees do not much resemble such research subjects. Our interviewees were much more like the "newsmakers" journalists interview. They were (and remain) relatively sophisticated adults not likely to be in awe of their interviewers. They were, for example, quite capable of asking to have the tape recorder turned off for a time when they wanted some part of the interview to be "off the record". For some of the interviewees, such a Bill Wulf, then

President of the National Academy of Engineering, it was the interviewers who were the ones likely to be in awe. Most of our interviewees were far enough along in their careers, and important enough in their discipline, to risk little should they happen to give offense. None was a graduate student or even an untenured assistant professor. They were not (in any significant way) dependent on the interviewers.

The First Amendment legally protects from IRB regulations the work of journalists based in a university in a way it does not protect other university-based researchers. Behind that legal protection is a moral argument balancing the need to protect the subjects of journalistic research against the importance of what may be gained by exempting the research from IRB regulations. I think the same moral argument applies to the research we did. What we could learn from the interviews was important; the vulnerability of our interviewees was low.

That is not to say that the sort of research we did should be entirely exempt from (formal) IRB review. Though our research group was (and remains) relatively conversant with research ethics (a majority having served on an IRB and several of us having as well a scholarly interest in the subject), we benefited from having the IRB look at our research plan. Researchers, even those with a strong commitment to research ethics, have a tendency to overlook or underestimate ethical difficulties in their own research. Having the IRB look over our research plan and make suggestions certainly helped to compensate for our (natural) partiality. The IRB identified questions we needed to think through (for example, how much confidentiality to provide documents). We might have benefited even more from IRB review had its regulations distinguished between our sort of research and the standard social-science research involving human subjects.

13.4.07. **Ordinary standards of confidentiality for medical documents should not (automatically) apply to "newsworthy" documents generated for another purpose.** We may divide the documents collected, placed in the archive, and used (or at least useable) during the writing of this book into at least six categories. For each category, the weight of reasons favoring confidentiality is different.

Part of the archive consists of articles published in a newspaper, magazine (including some sent out to subscribers by email), or online (for example, an IEEE-CS website). These, of course, need not be considered confidential at all. They may be cited in the way any public document may. Closely related to these are official reports widely distributed but not published (for example, some of Gotterbarn's reports to the Joint Steering Committee, distributed to a large mailing list beyond the Committee without any indication that they were to be kept confidential). I see no reason why these should not be treated as if they were public documents (even though, strictly speaking, they are not).

A third category contains the broadcast emails (without any indication that they were in any way confidential). These differ from official reports only in lacking a certain formality. Insofar as they lack formality, they are likely to be less careful in expression. They are more likely to include intemperate remarks, grammatical errors, and other signs of spontaneity. Gotterbarn plainly began thinking of the working group's list as "closed" (much as one might think of email circulated among employees in a firm). That was why, initially, he took care to prevent spamming (by not allowing automatic retransmission of any message sent to the list). While spamming turned out not to be a problem, getting people to use the list—and later the listserv—was. As he sought to make using the list (and listserv) easier, he incidentally opened it to personal messages not intended for broadcast and therefore had to remind users of the

distinction between replying to individuals and to "all" or to the listserv. Gotterbarn has given consent to use *his* broadcast emails freely. Mechler, Miller, and most of the others who contributed important broadcast emails have done the same. One important contributor to the list, Prinzivalli, is dead; we felt justified in presuming his consent to be quoted (especially since Gotterbarn considered his contribution to the Code important). We have therefore not had to decide the hard question whether the newsworthiness of this category of email justified quotation where consent was deliberately withheld or just negligently not granted.

I have not, please note, justified quoting these emails by appeal to Gotterbarn's initial notice (now apparently lost) that a research group at IIT would be monitoring SEEPP's communications as part of NSF-funded research. I have not appealed to that notice because I have come to think that the time between it and most of the emails was too great—and that, in any case, the notice seems not to have been given in a way likely to be noticed. While no one we interviewed objected to the use of his or her emails in writing this book, no one recalled the notice, even those who had been put on a SEEPP list in late 1993. Consent should not be supposed to last longer than the memory of it. I think it reasonable to suppose that the memory would last a few months, perhaps even a year, but no longer than that without reminders. Most of the private emails quoted in this book date from a later period. Hence, we have abandoned any appeal to "original consent".

Two other categories of email in our archive have forced us to face the question of using email where we do not have consent (but where it was not refused). One of these categories consists of emails to Gotterbarn (or Mechler) from individuals on one of Gotterbarn's lists who chose to write Gotterbarn (or Mechler), not the list. This is a disparate category. Often, the reason for not using the list seems to have been doubt that it was working, not a desire to keep the communication confidential. Sometimes the reason for not using the list was that Gotterbarn had not used it, for example, because he was some place where he could not use it. His respondent simply pressed "reply"—without thought of confidentiality. Gotterbarn occasionally forwarded emails of this sort to the list when he was again in position to do it, suggesting that he too did not suppose private communication necessarily implied confidentiality. None of these correspondents seems ever to have complained of the re-transmission.

Sometimes the reason for writing privately was that the subject discussed was not appropriate for the list, for example, an editor's response to Mechler journal submission. Sometimes it is not clear why the writer chose a private message rather than the list. For all we know, he may have used Gotterbarn's address, rather than the list's, because he (or his email program) recalled the one and not the other. Where we have quoted emails in this category and not had consent to do so, we have avoided giving the writer's name. We thought this sufficient to protect the author's rights (especially when the author in question was identified in Gotterbarn's email only by a common name and an expired—and opaque—email address).

Student contributions to the Code, whether going to the list or to Gotterbarn alone, posed a special problem. Some of the students (Kanko's) had no choice about contributing. They had an assignment and their instructor passed the results to Gotterbarn. These statudents certainly have a right not to have their names connected with what they wrote. But what they wrote was, I thought, too important not to quote. I have therefore quoted (or paraphrased) them—without giving a name (or any other information likely to reveal an individual's identity. Other students (such as those at Eau Claire) wrote at the suggestion of their instructor but (apparently) without any pressure to write or expectation of confidentiality. They would nonetheless probably be

surprised, and not necessarily pleasantly surprised, to find themselves quoted by name in a book. I have therefore also quoted them anonymously. But anyone who checks the endnote and knows the student from class should be able to figure out who he is. This seems harmless since what they wrote was intelligent, not something to be ashamed of even if they have changed their minds since—and they wrote it to a stranger without any promise of confidentiality.

The sixth category of email consists of "quotations". Some of these are true quotations, a few lines taken from a larger document, but most are whole documents the email program automatically placed below the response. Whether the original writer intended the email to be confidential, it is not confidential once quoted in a document not itself confidential. Yet, I did feel a residual duty to say as little about the source of the quotation as consistent with making an appropriate use of it. Where the author writes in an official capacity, especially where the source of the email is important, I have generally named the author. Otherwise, I have not.

In discussing quotation of email, it is easy to confuse two issues, copyright and confidentiality. I believe my quotations are all either "fair use" or covered by consent actually obtained. Though copyright has not been a problem, I could have avoided copyright issues altogether by paraphrasing rather than quoting directly. I have often preferred to quote directly to preserve the immediacy, pungency, or individuality of the original. I have paraphrased only when I saw no advantage to direct quotation. But, even if I had paraphrased every email, I would still have had much the same problem of confidentiality. Confidentiality concerns content as much as expression. People have a (limited) right to control to whom they reveal what they think—whether their exact words or merely the sense expressed.

13.4.08. **One way to compensate for treating this sort of research as journalism rather than "human subjects research" is to have an advisory board that includes representatives of those under study.** In anthropology, it is today common to create a committee of "the community" (a rural village, say) to oversee the study of that community. Such a committee is a way to maintain good relations with the community (by, for example, providing insight into the culture before the researchers commit some gross indiscretion). It is also a way to protect the community from the researchers (by, for example, working out in advance the rules for conducting the research). The community committee considers questions not likely to arise in any ordinary IRB. For example, the community committee might make it a condition of research that no lodge songs be taped, transcribed, or otherwise carried beyond the lodge. Because the community committee provides this sort of control over research, it can vouch for the researchers, easing their way into a community that might otherwise be much less cooperative.[31]

Our advisory board functioned in much the same way even though its membership was only in part drawn from the community under study. Having advisors helped in obtaining documents. Gotterbarn's contribution was, though enormous and irreplaceable, no surprise. We had built it into our original plan. Engel's contribution, though much smaller, was both important and a surprise (though, in retrospect, it seems we should have anticipated much of it). Engel understood the IEEE-CS in a way no one else on the committee did, especially the relations among various committees, certain IEEE terminology (for example, the difference within the IEEE between "IEEE standard" and "IEEE Standard"), and ways in which we might unintentionally give offense by some seemingly unimportant choice of words (for example, by referring to associate members as "mere associate members"). He also loaned me one hard-to-find report from his own files, reminded the advisory board of the larger institutional context in

which the process we were studying took place, and provided the name and phone number of sources from whom certain documents might be obtained.

Most important here, Engel—as well as Gotterbarn and Bernstein—spoke for the community under study concerning what emails (and other communications) were appropriate to quote in this book, what should only be paraphrased, and what should only be mined for information. They also helped us decide when it was appropriate to give the source's name. Their concern with confidentiality was not primarily with privacy (as it was for the IRB) but with not (unnecessarily) hurting the feelings of people or entire organizations with whom they might later have to work. Often, the way to satisfy their concerns was not, as at first I supposed, to suppress the quotation in question, but to enlarge it or otherwise provide more context. They did not seem to mind the light of day falling on what they or their colleagues had done; what they minded was having what they had done put in a false light. Only when the question of confidentiality concerned a private individual who wrote directly to another private individual (or to an office holder in some private capacity) did they express concern about confidentiality similar to that the IRB typically expresses.

13.4.09. **There is always another interpretation.** On January 6, 2004, Mechler sent me a dramatic reminder of how different interpretations of the same event can be. The story told here is of well-meaning people devoted to a good cause and working in ways more or less meeting standards of good conduct. Yet, after reading all but this chapter, Mechler posed six "ethical questions" about the conduct I described—and asking what I thought. What I thought was, "Here are six questions my readers should think about":

> 1. If one volunteers, whether recruited or not, shouldn't one give time to the project or resign?
> 2. If one is working on a project with a clear objective[,] should one's ideas rule how the work is done or what work is done or should the objective [rule the work]?
> 3. Is it ethical for Don to change the code without the original developers being involved?
> 4. Is it ethical to develop the code using a political model instead of an ethical model?
> 5. From the beginning to the end of your chapters it appears like the leaders want certain people, stack the membership, instead of volunteers. I know of at least two calls [in IEEE-CS publications] for volunteers and ACM probably had one also. How many volunteers were from these calls and how many hand picked? Or maybe the practitioners of SE didn't volunteer for the task. How many volunteers were there? You never give a count.[32] This point is related to question four [4] above, political or ethical?
> 6. Is it ethical to force a method on reality, method fit to reality vs reality fit to method? I have been trying for years, with papers and examples, to depict another method to use, especially in social issues but have had little success.

There is a popular saying: "Hindsight is 20/20". It asserts, in effect, that once we know that a decision turned out badly, we know not only what we should not have done but also what we should have done instead. The saying is wrong. For hindsight to be 20/20, we would have to know not only what we do know, how things turned out, but also how things would have turned out had people conducted themselves differently (for example, whether the Code would have

passed if Gotterbarn had followed an “ethical model” instead of a “political model”). We cannot know the outcome of choices not made without going back into the “history lab”, reprogramming the world, and watching the alternative history reveal itself. Unfortunately, we have no history lab. We are stuck with one set of events and our opinions about what-would-have-happened-if. Some of those opinions may, upon closer examination, prove untenable. Others, perhaps both Mechler’s opinion (that the process could have been more “ethical” and still succeeded) and Gotterbarn’s (that it was a near-run thing as it was and any less political document would have failed), might, upon close examination, turn out to be equally consistent with what we actually know (though at odds with each other). Seldom does history tell us who is right. Hindsight is not much better than foresight.[33]

NOTES

[1] Much of sections 13.1 and 13.2 have been published as “Eighteen Rules for Writing a Code of Professional Ethics”, *Science and Engineering Ethics* 13 (July 2007): 171-189.

[2] G. W. F. Hegel, *Philosophy of History*, trans. by F. Sibree (Colonial Press: New York, 1900), p. 6.

[3] Hegel, *Philosophy of History*, p. 26: “The history of the world is not a theater of happiness. Periods of happiness are blank pages in it…” George Eliot said it better (but a bit later): “The happiest women, like the happiest nations, have no history.” *The Mill on the Floss,* bk. V, ch. 4.

[4] For details on the importance of record keeping even in the early history of engineering, see my *Thinking like an Engineer*, esp. pp. 8-12. The past that engineering uses may well not count as “history” (strictly speaking) because it often does not take the form of a narrative. It is “history” only in the lose sense of “the past”.

[5] Karl Marx, *The Eighteenth Brumaire of Louis Napoleon*, in *On Revolution*, edited and translated by Saul K. Padover (McGraw-Hill: New York, 1971), p. 245: “Hegel remarks somewhere that all great world-historic facts and personages appear, so to speak, twice. He forgot to add: the first time as tragedy, the second time as farce.”

[6] See, for example, Eliot Freidson, *Professional Powers* (University of Chicago Press: Chicago, 1986), especially, Chapter 6 (“The Question of Professional Decline”). While I do not accept Freidson’s definition of profession (for reasons given in 1.3, I think his debunking in this chapter of the independent-consultant model of professions (and, in general, of the single line of development picture of professions) quite useful for freeing up thinking about professions and their codes.

[7] Gotterbarn\94-96 MISC\OPGUIDE.

[8] Kenneth Kipnis, "Toward a Code of Ethics for Pre-school Teachers: The Role of the Ethics Consultant", *International Journal of Applied Philosophy* 4 (Spring 1988): 1-10.

[9] There is some research on “heuristics” that might be relevant. Some of this research concerns the layout of texts generally, but not applied to codes; some, to corporate codes of ethics (that is, the codes of ethics to be used by employees most of whom lack the education and commitment of the typical professional). Such research, though suggestive for design of professional codes, is far from definitive. Yet, any claim to know “what works” would, presumably, have to rely on such research if it is to have any empirical basis at all.

[10] The exact wording for the ACM is, “This Code and the supplemental Guidelines were developed by the Task Force for the Revision of the ACM Code of Ethics and Professional Conduct [with the names following].” The exact wording for the software engineering code is

much the same: “This Code was developed by the IEEE-CS/ACM joint task force on Software Engineering Ethics and Professional Practices (SEEPP) [with the names following].”

[11] Email (Gotterbarn to Davis), October 27, 1999.

[12] Email (Gotterbarn to Davis), October 27, 1999.

[13] For a good example of what experts can do, see Wallace C. Koehler and J. Michael Pemberton, “A Search for Core Values: Toward a Model Code of Ethics for Information Professionals”, *Journal of Information Ethics* 9 (Spring 2000): 26-54.

[14] For examples of poor advice on writing codes of ethics, see Karim Jamal and Norman E. Bowie, “Theoretical Considerations for a Meaningful Code of Professional Ethics”, *Journal of Business Ethics* 14 (1995): 703-714. This article is but one in a whole issue on codes of ethics. Among these, I found only one that seems to offer good advice (for managers at least): Muel Kapein and Johan Wempe, “Twelve Gordian Knots When Developing an Organization Code of Ethics”, *Journal of Business Ethics* 14 (1995): 853-69.

[15] Gotterbarn has doubts about this recommendation: “Do we really want to repeat the nightmare of 4 different approval processes?” GotterbarnChapter12cmt (September 30, 2004). That question answers itself. But the conclusion I draw from that answer is not to use a simpler process for amendment than for original adoption. A simpler method threatens the legitimacy of the amendments. The conclusion I draw is that the process of adoption of the original code should also be relatively simple. Adopting a code of ethics should not be a nightmare.

[16] Margali Sarfatti Larson, *The Rise of Professionalism: A Sociological Analysis* (University of California Press: Berkeley, 1977), pp. 57-58.

[17] Larson, *Rise*, p. 6. This appears as a footnote—with an asterisk at her first use of “professional project”. The footnote is unique in the book; all citations and all other side remarks of this sort are given in endnotes. My guess is that the footnote was added at the last minute in response to an editor’s concern for what the ordinary reader might suppose.

[18] Consider Larson, *Rise*, again, p. 49: “a successful project of professionalization, one that comes close to attaining the goals of market monopoly, social status, and work autonomy, must be able to combine certain structural elements.” A “code of ethics” is not on her list of “structural elements”.

[19] That there is in this story not much evidence of self-interest at work in the debate over licensing does not *prove* that self-interest has no significant part. What the story I tell should do is *shift the burden of proof.* Absence of evidence is, after a reasonably diligent search, evidence of absence.

[20] Even in SEEPP’s original (1993) flurry of 29 volunteers, philosophers were as numerous as lawyers (with one each, Fodor and Phillips).

[21] The ACM committee had two philosophers (Gotterbarn and Johnson) but no lawyer.

[22] See, for example, Stuart Shapiro, "Degrees of Freedom: The Interaction of Standards of Practice and Engineering Judgment", *Science, Technology, and Human Values* 22 (Summer 1997): 286-316.

[23] While this argument seems sound, I should point out that the one perfect stranger we interviewed in our backyard, Sigut (when he was in Chicago for a convention), responded much as interviewees we traveled long distances to interview. We had, of course, met him at *his* hotel—and otherwise worked to serve his convenience. Perhaps that explains the similarity between that interview and those we did "on site".

[24] The new digital recorders, with memory in gigabytes, may finally have solved this problem. I say "may" only because I have not yet used one in an interview. Experience with other new technologies has taught me to wonder what the hidden disadvantages are. (How, for example, will I save what I have on my digital recorder in a form recoverable in five or ten years—after several changes of format and technology?)

[25] There are two exceptions. Steve Barber wrote out his answers in advance. We merely discussed them during an interview of over two hours (in a coffee shop on Wall Street), with me writing in clarifications as seemed appropriate. I then sent him the revised interview, which he corrected and approved. Dennis Frailey also wrote out his answers in advance—both to the original questionnaire and to supplementary questions I sent him later. Our interview (in a restaurant) still lasted well over two hours, with my notes adding many details to what he had written out in advance.

[26] While I cannot deny that spontaneous expressions might give insight into the individual interviewed and (through him or her) into the large context, I have doubts. How does one decide when carelessness is only carelessness and when it is something more? Psychology is far from an exact science even when a psychologist has weeks or months to evaluate a patient. How reliable could our judgments be when we had only two hours mostly busy with other matters—and only a few words to study later? A spontaneous expression might suggest a hypothesis, but the hypothesis would be no more than that without a way to test it. How do we test it—except by trying to include the insight in the narrative and seeing whether the narrative seems more informative in consequence? My experience here is that, generally, the narrative is no more informative in consequence.

[27] Interestingly, there is a disagreement something like this between the common law and civil law concerning "testimony". For the common law, testimony (ordinarily) consists of the oral statement made in court (with documents having to be "read into the record" before they can be considered evidence). Even the court transcript is simply a representation of the oral event (minus certain improprieties "stricken from the record"). For the civil law, on the other hand, testimony is (ordinarily) a written document submitted into evidence. Oral testimony in open court is more the exception than the rule.

[28] Minutes of SEA meeting of February 7, 2003, p. 4.

[29] Even if there are academic customs settling how historians or social scientists are to share such "raw material", the interviewee would have no idea of them—and it is the interviewee whose consent is supposed to be "fully informed". What the protocol does not say can only be supposed implicit if the ordinary interviewee can reasonably be expected to infer it.

[30] The advisory board suggested a very similar interview strategy, for example, questions such as: "To what extent was the development of a code profession-oriented? To what extent government-imposed? And to what extent oriented toward technical education?" (Minutes of SEA meeting of February 7, 2003, p. 2.)

[31] Alison Wylie, "Science, Conservation, and Stewardship: Evolving Codes of Conduct in Archaeology", *Science and Engineering Ethics* 5 (July 1999): 319-336.

[32] Sec. 4.1 does in fact have a "count" (from 1993-94 for SEEPP) and I did count additional names from succeeding years (for Gotterbarn's working group). These total fewer than sixty. The problem is that I do not have a *complete* count for SEEPP or even for the one working group that seems to have worked. The reason I do not have either is that no complete list of volunteers survives. Even combining all the lists that do survive would not yield a complete list. While Gotterbarn does not seem to have been trying to "stack the committee", he does seem to have continually tried to add to its membership, both in order to have the "right names" associated with it and to have help with its work. His record keeping seems not to have been equal to the efforts he made to recruit.

[33] Compare, for example, this question put by J. Barrie Thompson, "Any Real Progress, or Is It Just Politics and Turf Wars?" *Forum for Advancing Software Engineering Education* 11 (September 15, 2001) http://www.cs.ttu.edu/fase/v11n09.txt: "Why has the project, led by Don Gotterbarn et al, concerned with defining a Software Engineering Code of Ethics and Professional Practice, received such approval and been accepted by bodies outside the US while SWEBOK has led to such controversy?"