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		4
Chapter 1:	This History, Professions, and their Ethics	12

Part One: Slow Starts and Wrong Turns

Chapter 2:	Before SEEPP, 1968-1994	25
Chapter 3:	SEEPP Begins, 1994	44
Chapter 4:	Failing—by the book, 1995	68
Chapter 5:	Version 1, The Miracle of ’96	90
Chapter 6:	The High Politics of 1996	126

Part Two: 1997—Three Versions in One Year

Chapter 7:	Winter Whirlwind, Version 2.0	156
Chapter 8:	English Spring, Version 2a-2.1	194
Chapter 9:	Back in the USA, Version 3	237
Chapter 10:	Slogging toward “Version 4.DONE”	271

Part Three: Looking for Closure

Chapter 11:	The Long Process of Approval, 1998	312
Chapter 12:	End Game, Version 5.2, 1999-2000	354
Epilogue:	Lessons for Code Writers, Theorists, and Researchers	374

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: Anurag Kabra, Vishal Mehta, Vikram Modi, Alana Platt, Sushma Shankar, and Rahul Trehan, 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.