Sci Eng Ethics (2007) 13:171–189
DOI 10.1007/s11948-007-9000-2

ORIGINAL PAPER

Eighteen rules for writing a code of professional ethics

Michael Davis

Received: 4 August 2006 / Accepted: 29 October 2006 / Published online: 30 January 2007

Abstract Most professional societies, scientific associations, and the like that undertake to write a code of ethics do so using other codes as models but without much (practical) guidance about how to do the work. The existing literature on codes is much more concerned with content than procedure. This paper adds to guidance already in the literature what I learned from participating in the writing of an important code of ethics. The guidance is given in the form of ''rules'' each of which is explained and (insofar as possible) justified. The emphasis is on procedure.

Keywords Ethics · Code · Profession · Drafting · Conduct

While some scientific societies (such as the American Institute of Chemistry) have had codes of ethics for a long time, many have adopted a code only recently or are only now considering such adoption.[1] Many engineering societies in the United States, Canada, Australia, and other English-speaking countries have recently revised their code of ethics or are considering doing so. Scientific or engineering societies of many non-English speaking countries have recently adopted their first code of ethics or are now considering doing so.[2] We seem to be in the middle of a great age of code writing, not only a

[1] See, for example, the codes of: American Physical Society, 1991; American Mathematical Society, 1994; American Society of Biochemistry and Molecular Biology, 1998; American Association of Clinical Chemistry, 2003.

[2] I have noted the following examples in engineering recently: the College of Engineers of Chile; the Royal Flemish Society of Engineers; and the Royal Academy of Engineering–Brazil.

M. Davis (✉)
Center for the Study of Ethics in the Professions, Illinois Institute of Technology, Chicago, IL, USA
e-mail: davism@iit.edu

good time to think about how such codes are written but a good time to offer advice on how to write them. While taking a (small) part in writing one recent code, I realized how little is in print to help. I thought putting down what I learned might be useful.[3]

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 under the direction of the ad hoc Joint Steering Committee to Establish Software Engineering as a Profession.

That double approval was an important event in the history of professions. Software engineers design, construct, test, and maintain software. There are today at least a million of them around the world.[4] They are not only a new profession but a big one. That alone makes what they did important. But there were other respects in which what they did should interest anyone already interested in codes of ethics. Much of the work of writing the code was done by email, leaving an extensive ''paper'' trail that could be studied (something rare in code writing). The process was also unusually open, allowing people not members of the profession, including a philosopher like me, to participate in the process (and to observe). There is much to learn from that process, as much from its mistakes as from its successes. I have summarized what I learned in eighteen ''rules''. For each, I explain my reasons for the rule to make clear both the intent of the rule and the weight it should be given. I conclude with a ''postscript'' explaining how to use the rules.

1. Keep the writing of a code of ethics separate from any discussion of professional 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 professional 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,

[3] For a description of my part in the drafting (and my participation in some of the mistakes), see Davis [5]

[4] 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.

sufficiently low that not adding to the obstacles seems prudent. For example, the ad hoc Joint Steering Committee lasted 7 years. It was replaced by a ''permanent'' coordination committee that lasted just over a year. The first avoided the subject of licensing. The second did a little with it. The ACM soon withdrew from the Coordination Committee, citing its interests in licensing, forcing the Coordination Committee to dissolve [4].

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 subject to counter-example—and therefore, apparently, false.[5] Many professions—such as university teaching, journalism, and financial analysis—are not licensed at all (and others, such as engineering and chemistry, are only partially licensed). The claimed definitional relation between licensing and profession has at least two practical implications inviting caution. First, at least some practitioners take theories seriously enough to vote against a code because, for example, theory tells them it is necessarily a step toward licensing and they oppose licensing. That is why it is wise to make clear that a code is not related to licensing in that way. Second, a code destined for use in licensing may have to allow some conduct that a code designed for individual guidance need not. Much of the difference between the two most important American codes of engineering ethics 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 more demanding ABET code was not.[6] Those convinced that a proposed code will be used in formal disciplinary procedures may oppose individual provisions, or even the code as a whole, though they would otherwise have supported it.[7]

2. 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 Standards writing procedure advises:

> The WG [Working Group] Chairperson should then identify an individual to author an initial draft. The author should be permitted to

[5] See, for example, Freidson [10], especially, Chapter 6 (''The Question of Professional Decline''). While I do not accept Freidson's definition of profession, 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) is quite useful for freeing up thinking about professions and their codes.

[6] This body, once known as ''the Accreditation Board for Engineering and Technology'' is now officially ''ABET, Inc.'' By historical accident, ABET, originally (and still) engineering's chief accreditation agency in the U.S., has become responsible for developing, maintaining, and revising engineering's most successful (or, at least, best known) code of ethics.

[7] Compare Association of American Medical Colleges [2]. This 42-page pamphlet has much to say about the substance of a code of research ethics, but only pp. 6–8 are about procedure.

> prepare the draft with or without additional input or assistance from any other WG members, at his or her discretion.[8]

Every code of ethics needs what one leader of the drafting process called ''a Thomas Jefferson'' to do a first draft. The drafter has two problems that need to be solved before many 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 10; its core should be no more than 3 or 4. The actual drafting should be one individual's work (though there might, for example, be one drafter at an early stage and another later). Early *criticism* should be the work of half dozen or so (and no more than 10).

3. Keep preparation for the first draft simple

The problem of content can be solved in several ways. One method is to: 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); and 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. (This is often called ''brain storming''.)

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) [17].

There are other ways to develop the *initial* content of a code. The exact way does not much matter because the *exact* content of 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 (or other organization) in question and that there is enough of it to get the drafter started. Form follows content.

The history of the Software Engineering Code nonetheless shows that not all procedures are equal. Indeed, that history definitely recommends against two strategies that might otherwise seem obviously worth pursuing. One strategy is ''divide and conquer'', breaking work on the code into parts; the

[8] IEEE-CS/ACM Software Engineering Code of Ethics Archive, Center for the Study of Ethics in the Professions, Illinois Institute of Technology, Gotterbarn\94–96 MISC\OPGUIDE.

other is choosing the code's form before drafting begins. These two strategies share a common mistake. They assume that we know more about the future code than we can know. Dividing the code into parts assumes we know what the parts of the code will be. A comparison of original division of working groups by the Software Engineering Ethics and Professional Practice Task Force (SEEPP) 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. (There is almost no overlap.) The main problem with choosing the form before the content is not that some form (say, a short code) may rule out a good deal of content (although it certainly may). The problem is that the form of a code should serve its content. Debating the form without knowing the content is likely to be uninvitingly abstract, the domain of a few monomaniacs. Our working group was saved from that abstract debate only because no one cared enough about the form to respond to our chair's call for discussion of ''the Scope''. Some attempts to write a code have collapsed because early disputes about the form the code was to take made participants despair about ever agreeing on significant content [3].

4. Get a draft as soon as possible *but do not circulate it to any authoritative body or authoritative individual until the draft is "final"* (that is, ready for submission to the body or bodies charged with adoption)

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''. 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.[9]

[9] For an illustrative violation of this rule, see Anderson [1]. We too violated it—with almost disastrous consequences (the rejection of the code and the dissolution of the task force).

5. 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 our working group was set up, the Steering Committee chair advised following the IEEE Standards procedure. That IEEE procedure is notoriously complex, yet it was much 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, 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 the 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). (Governing bodies generally have more to do than time to do it—and therefore have a tendency to ignore any committee that seems to be ''producing''.) The drafting committee should stick to the procedure (so long as not disapproved)—unless it discovers it has made a serious mistake. Most of the remaining ''rules'' concern this public procedure.

6. 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. As I look back on the six years from the start of writing the Software Engineering Code to its final adoption, I am still surprised at how often prominent individuals in software engineering seem to have misjudged the opinions of those they tried to speak for (for example, predicting widespread opposition to provisions the membership later approved by as much as 20 to 1). Second, the openness protects the drafting committee not only from the eccentricities of those outside the committee but from the tendency of drafting committees to forget practical constraints (always a tendency in so high-minded an enterprise as drafting a code of ethics). The Software Engineering Code went through five major versions (and several minor ones) before final adoption. High-minded individuals put many ''aspirational'' provisions into Version 2 or Version 3 only to find the membership of IEEE-CS and ACM editing them out when polled in preparation for

Version 4.[10] Presenting drafts at sessions of professional meetings, taking comments, and trying to revise in response before the next professional meeting, is one way to open the procedure. A survey published in appropriate professional journals is another. Surveying professional *students*, if they are far enough along, is yet another. 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's standard-setting process has an especially interesting way to open procedure. It is a system of *voting with comments.* This is a good way to open procedure in a large organization or collection of organizations. Indeed, the IEEE Standards procedure has at least three advantages over the procedures already discussed (even though it was designed for technical standards, not a code of ethics, and is notably harder to administer). 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 because the panel of several hundred ''experts'' asked to evaluate a draft is to be drawn from all significant constituencies. Second, the IEEE procedure, though only a sampling of the profession as a whole, 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. The writing of the Software Engineering Code began with a world-wide (and widely circulated) call for participants. That call brought in less than 60 names (out of the many millions theoretically eligible). A later survey, published in magazines with a combined subscription of perhaps a hundred thousand, brought in just over 60 ballots. The IEEE procedures brought in an initial vote of just under a 150 (much more than the other two methods combined). Apparently, people are more likely to participate in response to personal invitation than to 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 constituency). 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.

[10] The distinction between ''aspirational'' provisions and whatever the alternative is (''minimal'' or ''mandatory'') in fact proved unworkable for those drafting the Software Engineering Code of Ethics. The problem, I think, was that those involved in writing the code could agree on what could reasonably be asked of software engineers but not on what they should be told to think of doing beyond that minimum. The distinction between ''enforceable'' and ''unenforceable'' code also proved hard to maintain. There was no question that software engineers would *informally* enforce the code (for example, by pointing out to an erring member that she was violating a certain provision). The only kind of enforcement that was rejected was through formal disciplinary procedures, whether that of a professional association or a licensing body. The only distinction that seemed to make sense was that between ethical rules (enforceable by conscience or peer pressure) and law (or law-like rules) enforced by formal procedures.

Writers of codes are often advised to bring ''the public'', ''clients'' or other ''stakeholders'' into the drafting process. While there is in theory much to be said for such inclusiveness, there are at least three practical barriers. First, finding outsiders willing to participate in the lengthy process is hard. After all (as just noted), it is hard even to find insiders to do it. Second, the assumption is that the outsiders will know enough to offer useful suggestions. Whether they will probably varies a good deal with the profession in question. For example, the public may have more useful suggestions to make concerning the practice of lawyers or physicians than of actuaries or organic chemists. (I, a philosopher, initially had nothing of use to tell software engineers about what they should do to protect me.) Third, there is the problem of keeping outsiders outside. The process of drafting can itself be a process of acculturation. Where writing a code of ethics goes on very long, say, a year or even a few months, and a few ''outsiders'' are intimately involved from start to finish, the outsiders not only learn much about the profession but tend to think about the code more and more the way insiders do. Though the IEEE procedure seems designed to deal with all three problems, it actually offers a solution (rather than an accommodation) only to this third problem. It keeps outsiders away from the process until relatively late and then brings them in a way not likely to acculturate them. It deals with the first problem only by declining to inquire how much acculturation is necessary to get busy people to read, vote, and comment on the proposed code. It resolves the second by searching for outsiders who may be expected to bring some special insight to the code (which, of course, means outsiders who have at least one foot inside).

7. Email is no substitute for in-person meetings

Email was to be an important new tool to open the process of writing the Code to people who might otherwise not have participated. And it probably was. E-mail is much easier to use than phone or fax and much faster than conventional mail. It is cheap and there is no need to worry about time in distant time zones. Yet, 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 (in interviews I did as part of a study of writing the Code) themselves assigned to face-to-face meetings; even meeting by conference call was plainly ''second best''. Software engineers are just the people we might expect to be unduly partial to email (and all the other new technologies). Apparently, email is much more like phone or fax than like ''the gold standard'', meeting in person.

8. 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 deleting 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. The sudden wind that blows a code in may just as suddenly change direction and blow the code out. 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, soon be complaints, perhaps even within a year, if there is no schedule, just as there would be if work were plainly ''behind schedule''. Without an official schedule, the drafting committee is prey to every individual's sense of how long ''a thing like that'' should take.

9. Find ways to test the code by making people use it ("user testing")

This rule, 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, wide participation, or final approval, 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 a professional code user friendly rests on much more than anecdote or personal preference.[11] 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 get ''user feedback'', less expensive in time, is to use students in the appropriate professional program. Our experience with using students to test the Code is that they seem to have responded to it much as did mature practitioners. If the same is true in other professions, professional students could be a convenient stand-in for practitioners.

[11] For a good summary of the present state of research on codes of business ethics, see Schwartz [21] There is no similar body of empirical research for codes of ethics for professions, academic societies, or research.

What is actually needed is a systematic study of the format 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 user testing to identify serious impediments to using the code.[12] 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.[13]

10. Work for consensus (rather than simple majority)

Ideally, a code of professional ethics should consist of those standards everyone in the profession, at her rational best, wants everyone else in the profession to follow even if that means having to follow them too. Because people are seldom at their rational best, it is impractical to demand that everyone in a profession actually agree to the profession's 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 profession 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 survives by one vote today may die tomorrow with the change of a single vote. A code that has 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). The code's drafters are supposed to respond to the negative's reasons, either by explaining why they reject them or by revising the code in something like the way suggested. When the negative's suggestion is taken, the negative's vote automatically becomes positive (at least until the negative objects and puts forth a new reason for the

[12] There is some research on ''heuristics'' that might be relevant. (None of it considers codes of ethics as such.) Such research, though suggestive for design of codes of ethics generally, is far from definitive, especially for codes of professional ethics. Professionals are, in general, more educated than the average reader of a text. Yet, anyone claiming to know ''what works'' would, presumably, have to rely on ''user research'' with educationally appropriate users (or lack any empirical basis for claims about layout, structure, and vocabulary). For the current state of thinking on the language of codes (without any empirical testing), see Farrell and Farrell [7].

[13] Even when the question is the effectiveness of codes as such, we know very little (and what we know concerns codes of business ethics). See, especially, Weaver [25], Somers [23], Tucker et al. [24], and Weller [26]. In fact, most advice about codes concerns content. For a good sample of what is available, see Kultgen [19], Frankel [9], Fisher [8], Harris [12], Kapstein [15], and Schwartz [22]. The first four are concerned with professional codes; the last two, with business codes.

negative vote). Anyone designing a procedure for adopting a code should try for similar effects. The governing body should not find a decision to adopt the code hard. The process by which the code reaches the governing body should vouch both for the code's workmanship and for its support among the general membership. The life of a code of ethics lacking widespread support can be quite short.[14] The drafting process should build support.

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. 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 of the Software Engineering Code (the version adopted), though improved in many ways from Version 1, clearly suffers from not having a writer work it over just 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'').

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 leading to adoption of the Software Engineering Code. Hoping to involve a few hundred in the process of drafting, revising, and adopting a code of ethics may be reasonable; hoping to involve even a thousand probably is not. Much the same is true about other aspects of writing a 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.

[14] For three notable examples of short-lived codes, consider the code of ethics of the American Medical Association adopted in 1903 and replaced in 1912; the code of ethics of the American Bar Association adopted in 1970 and replaced in 1980; and the code of ethics that the IEEE adopted in 1987 and replaced in 1990.

13. Do not list "authors", "contributors", or the like in the code itself

The Software Engineering Code followed the ACM's code in listing names of individuals at its end, indicating that they were members of the body that developed the code.[15] For both, 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 objection is that including a list of names in the document adds to what one has to get through to use it (and to the cost of reproducing the document) 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 other 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 work. When I first asked the chair of the code drafting committee about his 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.[16]

Our chair 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? He could not (he answered) simply remove a name once he had put it on—even if he put it on only because the person named had once signed up for the work:

> 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 [names] appeared on all of the publications as

[15] 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]." http://www.acm.org/constitution/ code.html (accessed July 30, 2006). 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, including mine, following]." http://seeri.etsu.edu/Codes/TheSECode.htm (accessed July 30, 2006).

[16] Code of Ethics Archive, Email (Gotterbarn to Davis), October 27, 1999.

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.[17]

Our chair 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 part of the Code. He need only have followed standard practice, omitting all names of individuals from the Code itself.

14. Never suppose that there are experts on what a code should say

Over the 6 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 some time or other, 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 useful in developing the initial list of provisions or later thinking about how that list might be augmented.[18] 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 the others to conduct themselves. We have no procedure by which to learn what members of a profession will want at their rational best except by asking them what they do want, challenging those answers with reasons, and seeing whether their answers 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 are 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.[19]

[17] Code of Ethics Archive, Email (Gotterbarn to Davis), October 27, 1999.

[18] For a good example of what experts can do, see Koehler and Pemberton [18].

[19] For some poor advice on writing codes of ethics, see Jamal and Bowie [14]. Some of the advice is simply unhelpful, for example, ''To the extent possible the rules for professional conduct should protect, or at least not be inconsistent with, the public interest''. But some of the advice seems positively bad (because, for example, it seems to confuse a code with an academic paper), that is, ''the code should reference the controversial nature of the rule either through footnotes or an extended commentary justifying the inclusion of the rule''. p. 713. The Jamal–Bowie article is but one in a whole issue devoted to codes of ethics. Among all these papers, I found only one that seems to me to offer good (practical) advice (for managers at least): Kapein and Wempe [16].

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. 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 those highly respected in a profession generally are elsewhere, no professional 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 it but not successful enough to be insulated from its pedestrian concerns. They are also likely to be disproportionately academics, even in fields where academics are not otherwise prominent. Some of these academics may (like me) come from outside the profession itself.

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 the web has become an important means of publication. An organization that lacks its own web page may still place its code on other web sites. Among these, the largest by far specializing in codes of ethics is Codes of Ethics Online (ethics.iit.edu/codes), maintained at my home institution.

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 about how to interpret the code. These interpretations ("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 that something they have done is unethical—especially if they are so told by an organization the judgments of which their colleagues or employers are likely to hear of and take seriously.

17. Establish a procedure for review and revision

Like people, codes age. What was up-to-date in 1980 may sound ancient even 25 years later. Part of keeping a code a living 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 or repeal—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, or to a temporary committee established, say, every 5 or 10 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 the code's revision to final approval. The procedure should be, more or less, the same as the original procedure for adoption of the code or a clear improvement on it.[20]

18. Be wary of official translations of a 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 reappear all over again each time the code is translated into another language. Even members of the relevant profession 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

[20] For some interesting (if exotic) ideas for the review process, Goodpaster et al. [11].

undertake it. As the translations accumulate, revising the original code becomes more complicated than it would be if the code were in only one 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 web, should be revised as well. Translation creates a problem of coordination, one that grows as the number of translations grows.

19. Postscript: learning from history

Since I am claiming to learn from the past, I think I should explain how that is possible. That explanation will provide considerable guidance about how much weight to give my eighteen rules.

It is often said that ''those who do not learn from history are doomed to repeat it.'' Why ''doomed''? If history consisted entirely 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 Hegel, 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.''[21] The lesson of history is (it seems) that we do not learn from history; we are all doomed. Though sweeping enough to be widely quoted, Hegel's lesson seems too bleak to be true.

I do not deny that history (our rendition of the past) consists, in large part, of mistakes. How could it be otherwise? Humans often make mistakes and often find those of others interesting. That is why Hegel thought the history of a happy people (one for whom things generally go well) a ''blank''.[22] 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 (that is, tell 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

[21] Hegel [13], p. 6.

[22] Hegel [13], 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 (as we might expect, but a bit later): ''The happiest women, like the happiest nations, have no history.'' *The Mill on the Floss*, bk. V, ch. 4. [6]

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.[23]

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.[24] The difference between tragedy and farce is big 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 sad 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'' (even if equally important)? What can we take from one circumstance and apply in circumstances that always differ from it 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 claim to draw from ''history'' here.

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 should follow it. If following it ends up badly more than rarely, we should 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. The end of the world is too late to learn anything useful—and everything else seems too soon to learn anything definitively.

[23] For details on the importance of record keeping even in the early history of engineering, see Davis [4], 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 looser sense of ''the recorded past''.

[24] Marx [20], 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.''

The ''lessons of history'' are interpretations of events, interpretations that (as 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 (as I have stated them here), the lessons of history should be treated as advice: ''Think about this.'' That, anyhow, is the spirit in which I have offered my eighteen rules.

Acknowledgements Work on this paper was funded in part by a grant from the National Science Foundation (SES-0117471). I should like to thank the advisory board with which I worked under that grant for many helpful comments on early versions of this paper. I should also like to thank the reviewers for this journal.

References

1. Anderson, R. E. (1994). The ACM code of ethics: History, process, and implications. In C. Huff & T. Finhold (Eds.), *Social issues in computing: Putting computing in its place* (pp. 48–62). New York: McGraw-Hill, Inc. .
2. Association of American Medical Colleges (1997). *Developing a code of ethics in research: A guide for scientific societies*. Washington, D.C.: Association of American Medical Colleges.
3. Berleur, J., & d'Udekem-Geves, M. (2000). Codes of ethics: Conduct for computer societies: The experience of IFIP. In P. Goujon & B. Hériard Dubreuil (Eds.), *Technology and ethics: A European quest for responsible engineering* (pp. 327–350). Leuven, Belgium: Peeters.
4. Davis, M. (1998) *Thinking like an engineer*. New York, Oxford University Press.
5. Davis, M. (2000). Writing a code of ethics by e-mail: Adventures with software engineers. *Science Communication, 21*, 392–405.
6. Elliot, G. (1998) *The mill on the floss*. New York: Oxford University Press.
7. Farrell, H., & Farrell, B. J. (1998). The language of business code of ethics: Implications of knowledge and power. *Journal of Business Ethics, 17*, 587–601.
8. Fisher, C. B. (2003). Developing a code of ethics for academics—Commentary on 'Ethics for all: Differences across Scientific Society Codes' (Bullock and Panicker)''. *Science and Engineering Ethics, 9*(2), 171–179.
9. Frankel, M. S. (1989). Professional codes: Why, how and with what impact? *Journal of Business Ethics, 8*, 109–115.
10. Freidson, E. (1986). *Professional powers.* Chicago: University of Chicago Press.
11. Goodpaster, G. E., Maines, T. D., & Weimesrskirch, A. M. (2004). A Baldrige process for ethics?. *Science and Engineering Ethics, 10*, 243–258.
12. Harris, C. E. (2004). Internationalizing professional codes in engineering. *Science and Engineering Ethics, 10*, 503–521.
13. Hegel, G. W. F. (1900). *Philosophy of history*, trans. by F. Sibree. New York: Colonial Press.

14. Jamal, K., & Bowie, N. E. (1995). Theoretical considerations for a meaningful code of professional ethics. *Journal of Business Ethics, 14*, 703–714.
15. Kapstein, M. (2004). Business codes of multinational firms: What do they say? *Journal of Business Ethics, 50*, 13–31.
16. Kapein, M., & Wempe, J. (1995) Twelve Gordian knots when developing an organization code of ethics. *Journal of Business Ethics, 14*, 853–869.
17. Kipnis, K. (1988). Toward a code of ethics for pre-school teachers: The role of the ethics consultant. *International Journal of Applied Philosophy, 4*, 1–10.
18. Koehler, W. C., & Pemberton, J. M. (2000). A search for core values: Toward a model code of ethics for information professionals. *Journal of Information Ethics, 9*, 26–54.
19. Kultgen J. (1983). Evaluating codes of professional ethics. In W. L. Robinson & M. S. Pritchard (Eds.), *Profits and professions. Essays in business and professional ethics* (pp. 225–263). New Jersey: Humana Press.
20. Marx, K. (1971). The eighteenth Brumaire of Louis Napoleon. Edited and translated by S. K. Padover, *On revolution*. New York: McGraw-Hill.
21. Schwartz, M. (2001). The nature of the relationship between corporate codes of ethics and behaviour. *Journal of Business Ethics, 32*, 247–262.
22. Schwartz, M. S. (2005). Universal moral values for corporate code of ethics. *Journal of Business Ethics, 59*, 27–44.
23. Somer, M. J. (2001). Ethical codes of conduct and organizational context: A study of the relationship between codes of conduct, employee behaviour and organizational rules. *Journal of Business Ethics, 30*, 185–195.
24. Tucker, L. R., Stathakopolous, V., & Ch, P. H. (1999). A multidimensional assessment of ethical codes: The professional business association perspective. *Journal of Business Ethics, 19*, 287–300.
25. Weaver, G. R. (1995). Does ethics code design matter? Effects of ethics code rationales and sanctions on recipients justice perceptions and content recall. *Journal of Business Ethics, 14*, 367–285.
26. Weller, S. (1988). The effectiveness of corporate codes of ethics. *Journal of Business Ethics, 7*, 389–395.