The Software Engineering Code of Ethics and Professional Practice has recently been approved. This article looks at the immediate and long-term implications: Why does a profession need a code of ethics? How will this code function in an emerging profession like software engineering? What impact will it have on software practitioners?

How the New Software Engineering Code of Ethics Affects You

Don Gotterbarn, Software Engineering Ethics Research Institute

According to the Texas Board of Licensing, software engineering is "a distinct discipline under which engineering licenses can be issued."[1] What does this mean to practicing software engineers? What are the positive aspects of this choice to become a profession? How can we exploit this movement toward professionalism in a positive way? There are many as-yet unanswered questions.

One example of this movement toward professionalism is the recent adoption by the IEEE Computer Society and the Association for Computing Machinery of a Software Engineering Code of Ethics and Professional Practice http://www-cs.etsu.edu/seeri/secode.htm). How does this new code fit into the picture? The Code describes the ethical and professional obligations against which peers, the public, and legal bodies can measure a software developer's behavior. The recent debates about the content of the Code, prior to adoption, reflected tensions that needed to be resolved between current and developing standards. Should the Code be interpreted as a legal document or as a document intended to inspire good

0740-7459/99/$10.00 © 1999 IEEE

practice? Can the Code be used to guide professionals in their decision-making during software development? Or to alert practitioners to those things for which they are accountable? Understanding how and why the Code was developed will help us anticipate and plan for the potential consequences of our decisions.

The development of the Code was an international project with participants from every continent. Major companies helped by posting early drafts on their bulletin boards for comment by employees. The draft Code was reviewed by members of several professional computing societies and went through several revisions (see the "Chronology of the Code" sidebar for details). Because of the way the Code was developed, it is not unreasonable to say that this Code represents a movement toward an international consensus of what software engineers believe to be their professional ethical obligations.

Chronology of the Code

Draft 1: delivered to IEEE-CS/ACM Steering Committee December 1996.

Draft 2: widely circulated for comment January through March 1997. Published in SIGSOFT and SIGCAS bulletins in July 1997

Draft 3: circulated to industry and other professionals and then published with a turnaround ballot in November 1997 *Computer* and *Communications of the ACM*.

Draft 4: revision based on comments and ballots regarding version 3. Submitted to Steering Committee in December 1997.

Draft 5: passed a complete IEEE formal technical review process in September 1998.

Draft 5.2: passed legal review.

Draft 5.2: approved by the ACM November 1998 and the IEEE Computer Society December 1998.

Why Our Own Code of Ethics?

The simple answer to this question is that most other professions operate under explicit ethical standards stated in profession-unique codes of ethics. Because professionals have an enormous impact on the lives and well-being of others, they and those they affect have set higher and broader standards of practice for professionals than is expected of nonprofessionals.

Some codes seem trite or irrelevant because they merely state high-sounding value claims and speak only to a single level of professional obligation. A professional code actually needs to address three levels of obligation. The first level is a set of familiar ethical values, such as integrity and justice, which professionals share with other human beings by virtue of their shared humanity. The second level obliges professionals to more challenging obligations than those required at the first level. By virtue of their roles and special skills, professionals owe a higher order of care to those affected by their work. Code statements at this level express the obligations of all professionals and professional attitudes. The third level

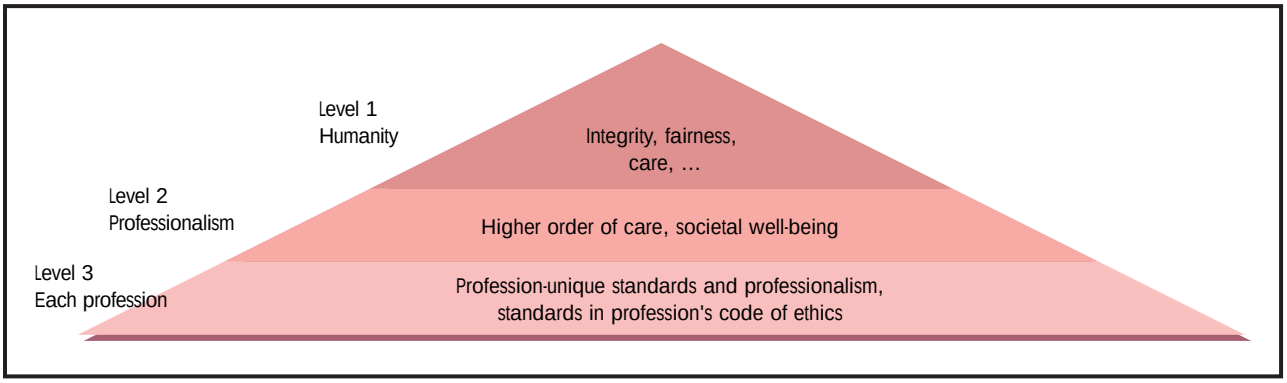

Figure 1. The cumulative levels of professional obligation. While all people share the obligations at Level 1, professionals also carry the responsibilities expressed in Level 2. A member of a particular profession might have additional, unique obligations, as articulated in Level 3.

includes all the obligations of the first two levels along with several obligations that derive directly from elements unique to the particular professional practice. For software engineering, code statements at this level assert more specific behavioral responsibilities that are closely related to current best practices—for example, "3.10 Ensure adequate testing, debugging, and review of software and related documents on which they work." Figure 1 shows the hierarchy and cumulative nature of these levels.

Professional codes that merely address Level 1 do not help an emerging profession like software engineering clarify expectations and appropriate behavior of professionals. Software Engineering's conscious and public choice to become a profession commits it to all three levels of professional/ethical obligation.[2] The short statement of the Code (see "The Code: A Short Version" sidebar) is an abstraction from the complete Code that addresses all three levels of professional obligation. It is because of the third level of obligation that individual professions need their own profession-unique codes.

But why should a profession have a code of ethics at all? A code can simultaneously serve several functions.

♦ It might be designed to be inspirational—either for "positive stimulus for ethical conduct by the practitioner"[3] or to inspire confidence of the customer or user in the computing artifact and confidence in its creator.[4] Unfortunately, inspirational language tends to be vague, limiting the code's ability to help guide professional behavior.

Sanctions will occur only when the Code is publicly adopted as a generally accepted standard of practice.

♦ Historically, there has been a transition away from regulatory codes, designed to penalize divergent behavior and internal dissent, toward more normative codes, which give general guidance. Although a professional can use a normative Code to examine alternative actions, such codes are only a partial representation of a profession's ethical standards.[4] Because the use of normative codes requires moral judgment on the part of the professional, they should not be considered a complete procedure for deciding what is right or wrong.

♦ Codes also serve to educate both prospective and existing software engineers about their shared commitment to undertake a certain level of quality in their work and their responsibility for the well-being of the customer and user of the developed product.

Codes also serve to educate managers (see Principle 5 of the full version of the Code, http://computer.org/tab/swecc/code/htm) of software engineers, and to educate those who make rules and laws related to the profession, about expected behavior. Managers' and legislators' expectations will affect what is asked of software engineers and what laws are passed relating to software engineering, respectively. Directly and indirectly, codes also educate management about their responsibility for the effects and impacts of the products developed.

Codes also indirectly educate the public at large about what professionals consider to be a minimally acceptable ethical practice in that field, even as practiced by nonprofessionals.

♦ Codes provide a level of support for the professional who decides to take positive action. An appeal to the imperatives of a code can be used as counterpressure against others' urging to act in ways inconsistent with the Code.

♦ Codes can be a means of deterrence and discipline. They can serve as a formal basis for action against a professional; for example, some organizations use codes to revoke membership or suspend licenses to practice. Because codes usually define in detail the minimal behavior for all practitioners, the failure to meet this expectation can be used as a reasonable foundation for litigation.

♦ Codes have been used to enhance a profession's public image. They prohibit public criticism of fellow professionals, even if they violate some ethical standard.

The specific functions selected for a code affect its potential impact. Codes designed for an emerging profession, including Software Engineering, emphasize education, guidance, support, and inspiration. Our Code does not specifically address deterrence and discipline or include standards for prohibiting someone from practicing software engineering.

Can you lose your job because you did not follow our Code? Software Engineers are not required to belong to a single professional organization in order to practice, and there are no realistic sanctions that can be imposed for code violations. Sanctions will occur only when the Code is publicly adopted as

a generally accepted standard of practice, and when both society and legislators view the failure to follow the Code as negligence, malpractice, or just poor workmanship. The Code clearly defines the responsibility of the profession and the professional to promote and protect positive values. For this type of sanction to work, professionals and nonprofessionals must be educated about the professional obligations described in the Code.

Professional Tensions

There were at least three distinct controversies during the Code's development. From least to most significant respectively, the controversies were about the approach to ethics; discomfort with strong, powerful, and incomplete ethical imperatives; and divergent views about software engineering technical standards.

Approaches to ethics

Surprisingly, the major tension centered not around technical issues but rather on two distinct approaches to ethics: *virtue ethics* and *rights/obligations ethics*.[6] Virtue ethics holds the optimistic view that if people are simply pointed in the right direction, their moral character will guide them through ethical problems. Reviewers from this camp wanted a Code that was mostly inspirational, with minimum detail. They put a heavy emphasis on the autonomy of a professional's judgment. The other position—rights/obligations theory—consists of spelling out precisely one's rights and responsibilities. Believers in this theory wanted a very detailed Code. For example, one reviewer wanted the Code to include a standard of measurement for each imperative—to state exactly how many tests need to be done to ensure adequate testing. The Rights/Obligations folks used a legalistic model to evaluate each imperative. The problem was that any imperative acceptable to one group was not acceptable to the other. The Code addresses this significant tension in a number of ways.

To address the problem of insufficient guidance for decision making, the Code incorporates some directions for ethical decision making in the preamble and an acknowledgment that the Code's normative premises should not be read as complete descriptions or legalistic statements. The Code aids decision making by overcoming two difficulties with other codes. First, most codes of ethics provide

The Code: A Short Version

Software Engineering Code of Ethics and Professional Practice

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

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.

The full version of the Code is available at http://www-cs.etsu.edu/seeri/secode.htm and http://computer.org/tab/swecc/code.htm#full.

a finite list of principles that are often presented as a complete list; readers might presume that only the things listed should be of ethical concern to the professional. Second, many codes provide little (if any) guidance for situations where rules having equal priority appear to conflict. This equal priority leaves the ethical decision-maker confused. The Software Engineering Code addresses both of these limitations.

Several companies have already adopted the Code, and its principles have been incorporated into their standards of practice.

The Code does not leave the reader without support. Here are some of its suggestions about how to make decisions:

> *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.*

This section also provides guidance in selecting between apparently conflicting principles in the Code by stating that the primary concern in all decisions must be public well being, as opposed to loyalty to the employer or the profession.

The Code addresses the rights–virtue tension in another way—through its structure. The Code is organized around eight broadly based themes. Under each theme or principle is a series of clauses giving examples of how that theme applies to software engineering practice. Thus, under "Principle 1, PUBLIC, Software engineers shall act consistently with the public interest" is the illustrative clause 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 the software or related documents."

Discomfort with the rules

Some reviewers' discomfort was founded in perceptions of powerlessness. These people reacted to a few Code principles with "That is a good idea but…." For example, some claimed that Clause 3.05, "Ensure an appropriate methodology for any project on which they work or propose to work," was an unrealistic demand; they could not give such assurances unless they were project manager. There was also considerable debate about Clause 3.06—"Work to follow professional standards, when available, that are most appropriate for the task at hand, departing from these standards only when ethically or technically justified." Some were concerned that the precise standard was not (and could not be) specified (see the following section). Some wanted to reject this clause because they thought that being the first to market—gaining economic advantage—justified abondoning standards. The tension between competing best practices was another concern. The fear of a legalistic interpretation of the Code's normative principles also caused concern.

It is important to note that many of the reviewers moved in an opposite direction. They wanted to strengthen the rules and close loopholes. For examples, Clause 6.08 was "Take responsibility for detecting, correcting, and reporting *significant* errors in the software and associated documents on which they work." Most reviewers advocated removing the word "significant" because they did want to leave room for someone to claim, "I found lots of errors, but I didn't think any of them significant." This kind of strengthening of principles was fairly common. For example, 1.04 requires that a software engineer disclose any dangers created by the software. Reviewers inserted the phrase "actual or potential dangers" to prohibit someone from not reporting a danger because it was not yet real. They also changed the expression " Be fair and truthful" to "Be fair and avoid deception" in all statements, particularly public ones, concerning software or related documents (clause 1.06). This not only prohibits falsehoods but also covers the case of not disclosing relevant information, deception by omission.

Interaction between technical and ethical standards

On several occasions, someone asked the task force to put specific standards or best practices into the Code, for example, "Path testing must be done

on all software with a cyclomatic complexity greater than 12." We decided against this, because we were particularly sensitive to the relationship between ethical and technical software engineering standards. We considered some of these relationships to be obvious: when an organization accepts technical standards, it routinely follows them to improve the quality of work. It would be unethical (and in some cases legally negligent) *not* to follow these standards carefully. In other situations, software engineering "standards" and best practices compete with each other. The Code says choose from among the competing best practices.

As our discipline gains knowledge, our standards improve. Next year's competing best practices will be different from this year's. Nevertheless, the Ethics Code accommodates this dynamic aspect of science and software engineering. If we had included specific standards, the Code would likely be obsolete the moment it was approved, because techniques are constantly improving and because the Code's revision process is at least a year long. Moreover, including some standards and excluding others could lead to a premature "blessing" of some standards, encouraging a harmful retardation of the development of better software development techniques. Instead, the Code refers to "currently accepted standards" in the hope that, as standards are increasingly well defined and accepted, the Code can encourage their use.

A Common Commitment

In spite of these controversies, software engineering professionals understand the significance of their work and their ethical obligations to their products' stakeholders. As the Preamble to the full version of the Code states,

> *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.*

There was near unanimity that software engineers must behave proactively when they are aware of potential difficulties in a system. Several clauses in the Code require preemptive reporting of potentially dangerous situations, and clauses 6.12 and 6.13 outline a procedure for whistle blowing.

Early Adopters

Several major organizations have already adopted the Code or have incoroporated its principles into their standards of best practice. They include

- Construx Software, Bellevue, Washington
- Institute for Management Information Systems, United Kingdom
- Monmouth University SE training for the US Army Communications and Electronic Command Software Engineering Center, Fort Monmouth, New Jersey
- Pointer Software Systems, Israel
- Siemens Information Systems' Software Development Center, Bangalore, India
- United Kingdom Royal Mail

Your organization is invited to join the growing number of organizations that have adopted the Code. Adoption of the Code will publicly endorse your companies' commitment to quality and good practice, and endorsement will serve the profession at large by promoting a universal standard of practice. For more information, contact the Software Engineering Professional Ethics Project at SEPEP@etsu.edu.

Because some people wanted a concise, relatively high level, inspirational code and others wanted a more detailed document that would guide practitioners in making technical decisions ethically, we compromised. The Code comes in two versions: the short document fulfills the desire for conciseness, and the long one offers more details.

The Code received a consistently high level of agreement about the behavior expected of a professional software engineer. Software engineers generally agree about their obligations. However, it was clear from the comments that various forces pressure software engineers to not always fulfill these obligations. We have also been gratified at the level of interest from the business sector. Several major companies have already adopted the Code (see the "Early Adopters" sidebar), and its principles have been incorporated into their standards of practice. The Code of Ethics and Professional Practice has also been included as part of employment contracts that are signed at the time of employment.

A code fulfilling its educational function will change the approach of many to software development. The adoption of the Code by multiple

societies and businesses moves it toward a profession's code rather than that of an individual professional society. As the Code is publicized and continues to gain support from other professional societies and corporations, it will become a de facto standard. Just as several potential customers have decided that they don't want to do business with CMM Level 1 companies, so people eventually will decide that they will not do business with those who do not follow the Code. As the Code becomes accepted, it will provide counter-pressure against those who ask their staff to behave unprofessionally and unethically. If nothing more, it will help many become aware that their behavior might be unethical. The existence of this standard will make them pause and think about the potential effects of their actions.

The Software Engineering Code of Ethics and Professional Practice is a useful tool. It educates and inspires Software Engineers. The Code instructs practitioners about the standards that society expects them to meet and what their peers strive for and expect of each other. The Code offers practical advice about issues that matter to professionals and their clients, and it serves to inform policy makers about ethical constraints imposed on software engineers.

The Code indirectly educates the public at large about the responsibilities that are important to and accepted by the profession—what Software Engineers consider to be minimally acceptable practice—even when a nonprofessional practices it. Thus the code can be a catalyst to simultaneously raising the internal expectations of a profession and the expectations of the society at large.

The Code is a dynamic document, a method for education, inspiration, and continued study and debate (see the "Work in Progress" sidebar). It provokes serious discussion about the software engineering discipline, its responsibilities, and its future. The Code directs us to be a part of that future as we improve our profession and ourselves. ❖

A Work in Progress

The Computer Society and the ACM continue to support this effort and have recently formed a Software Engineering Professional Ethics Project under the auspices of the Software Engineering Coordinating Committee. This project's tasks include promoting the adoption of the Code by industry and other professional bodies as well as developing education materials and case studies to help aspiring software engineers understand the obligations of a practicing professional. Those interested in participating should contact sepep@etsu.edu.

References

1. Texas Board of Professional Engineers, "Board Establishes Software Engineering Discipline," http://www.main.org/peboard/sofupdt.htm (current 22 Oct. 1999).
2. D. Gotterbarn, "Software Engineering Ethics," *Encyclopedia of Software Engineering*, J. Marciniak, ed., John Wiley & Sons, New York, 1994.
3. M.W. Martin et al., *Ethics in Engineering*, 2nd ed., McGraw-Hill, New York, 1989.
4. R. Anderson, "The ACM Code of Ethics: History, Process, and Implications," *Social Issues in Computing*, McGraw-Hill, New York, 1995, pp. 48–72.
5. D. Gotterbarn, "Software Engineering: The New Professionalism," *The Professional Software Engineer*, C. Myer, ed., Springer-Verlag, New York, 1996.
6. S.L. Edgar, *Morality and Machines: Perspectives on Computer Ethics*, Jones and Bartlett Publishers, Sudbury, Mass., 1997.

About the Author

Don Gotterbarn is director of the Software Engineering Ethics Research Institute and a professor of computer science at East Tennessee State University, where he helped develop a master's of software engineering curriculum. His research has focused on performance prediction for a distributed Ada compilation, object-oriented testing, software engineering education, and software engineering ethics. Under a US National Science Foundation grant he is currently developing an ethics audit decision support tool for project management.

Gotterbarn holds a PhD from the University of Rochester. He chaired the Joint IEEE Computer Society–ACM Task Force on Software Engineering Ethics and Professional Practice and is the vice chair of Computers and Society. Contact him at the Software Engineering Ethics Research Inst., East Tennessee State Univ., Box 70,711, Johnson City, TN 37614-0711; gotterba@etsu.edu, http://www-cs.etsu.edu/gotterbarn.