ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/234825637

Computer ethics in the undergraduate curriculum: case studies and the joint software engineer's code

Article *in* Journal of Computing Sciences in Colleges · December 2004

CITATIONS
11

READS
3,529

2 authors:

Donald Gotterbarn
East Tennessee State University
176 PUBLICATIONS **1,414** CITATIONS
SEE PROFILE

Keith Willam Miller
University of Missouri - St. Louis
227 PUBLICATIONS **4,639** CITATIONS
SEE PROFILE

Some of the authors of this publication are also working on these related projects:

ACM Code of Ethics revision View project

Rise in Autonomous Weapons Systems: Is it a Security or Threat? An ethical analysis! View project

All content following this page was uploaded by Donald Gotterbarn on 09 April 2016.

The user has requested enhancement of the downloaded file.

COMPUTER ETHICS IN THE UNDERGRADUATE CURRICULUM: CASE STUDIES AND THE JOINT SOFTWARE ENGINEER'S CODE*

Dr. Don Gotterbarn
Eastern Tennessee State University
Dept. of Computer Science
Box 70711
Johnson City, TN 37614

Dr. Keith W. Miller
University of Illinois at Springfield
Dept. of Computer Science
One University Plaza
Springfield, IL 62703

ABSTRACT

This paper illustrates how to use the Software Engineering Code of Ethics and Professional Practice [1,2] in three case studies suitable for computer science instruction. This code of ethics was approved by both the Association of Computing Machinery (ACM) and the IEEE Computer Society in 1998. Since then, the code has been translated into seven more languages, and adopted by organizations in many countries.

The paper argues that instruction in ethics is vital in computer science education, and that case studies featuring the Software Engineer's Code can be an effective method for that instruction. The three cases all focus on realistic situations in which a software engineer must make choices that involve technical and ethical judgments. For each case, the paper identifies relevant sections of the Code, and analyzes the case study using ideas from those sections.

1. INTRODUCTION

As society becomes more dependent on computing, there is pressure to increase the emphasis on professional ethics in college curricula. Sometimes this responsibility is assigned to professors of philosophy who understand significant ethical issues but are ill

---

* Copyright © 2004 by the Consortium for Computing Sciences in Colleges. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CCSC copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Consortium for Computing Sciences in Colleges. To copy otherwise, or to republish, requires a fee and/or specific permission.

equipped to deal with issues that depend on subtle technical details of computing. We contend that it is essential to discuss professional ethics in technical classes taught by faculty competent in computer science. We further hold that one effective technique is case analysis based on the Software Engineering Code of Ethics and Professional Practice. Referred to as "the Joint Code" or simply "the Code" in this paper, the Software Engineering Code of Ethics and Professional Practice can be seen in its entirety (including its short form and a more detailed, longer form) at its website [1]. Information about the adoption of the Code by the ACM and the IEEE Computer Society is included in [2]. Figure 1 shows the short form of the Code.

There is little contention about the effectiveness of using cases to train practitioners in many professions. [3] But there are numerous questions about using a code of ethics to fill the critical role of support and insight for professional ethics. On the negative side, codes often seem to be too general. Their scope does not seem to relate uniquely to the domain of software development project. Even if you can identify imperatives that need to be followed, these imperatives may seem to conflict, leaving the developer waffling between two alternatives. Worst of all from the teaching perspective, codes of ethics may seem dry and boring to students.

But there are good reasons to use a code of ethics, especially if it can avoid some of the drawbacks mentioned above. Codes of ethics have been developed to educate the public, practitioners, and students about the obligations of software professionals. A good code of ethics for software engineers can guide all professional software developers, not just bad actors. In what follows we demonstrate the utility of using one particular code, the Joint Code introduced above, using three case studies that can be used in computer science classes.

In this paper, we concentrate on using the Joint Code as it relates to practicing professionals and to students who aspire to joining the profession. Indeed, the Joint Code was developed with those audiences in mind. [2]

1.1 A Short History of the Joint SE Code

The ACM - IEEE Computer Society Joint Steering Committee for the Professionalization of Software Engineering established the Software Engineering Ethics and Professional Practice task force to document and codify software engineering's standards of ethical and professional practice. The development of the Joint Code was an international project with participants from every continent. The participants responded to a call for participation sent to the memberships of the ACM and the IEEE Computer Society, news groups, other professional societies, companies, and interested parties. Major companies aided in the process by posting early drafts of the Code on their electronic bulletin boards for comment by their employees. Reviews, re-drafts, and balloting on the Code were conducted in the international arena.

Members of the task force formulated imperatives that were refined over the long history of the code's development. The draft Code was reviewed by members of several professional computing societies and went through several revisions. Version 3 appeared with a turnaround ballot in the journals *IEEE Computer* and *Communications of the ACM.* Most clauses in the code received better than a 90% approval rating. Contributed

comments led to the development of Version 4 which was submitted for peer review using the IEEE's formal technical standard review process. The Code passed this process and comments from that review were used to develop the Version 5.2 of the Code. Version 5.2 was approved by the ACM in November of 1998 and by the IEEE-Computer Society in December of 1998. Because of its wide dissemination (the Joint Code has been translated into Chinese, Croation, French, Hebrew, Italian, Japanese, and Spanish), and its growing influence, it is not unreasonable to say that this Code represents movement toward an international consensus of what software engineers believe to be their professional ethical obligations.

1.2 Why the Joint SE Code is Useful for Future Computing Professionals

We want our students to understand their obligations to the well-being of those who use their software, and the Code stresses this obligation. Because computer science education emphasizes technical information and skills, students may fail to recognize when they encounter a significant ethical issue. The Code can help them identify these situations.

Our students may someday work in environments where they are asked to act in unprofessional ways by their management, and the Code will help them withstand those pressures. Many of our students will become managers; the Code includes clauses that explore management obligations, and that will help students prepare to fill those management roles ethically.

In general, the Code will help students think carefully about their ethical obligations as it encourages them to consider the consequences of their actions. The Code also alerts students that unethical actions are condemned by professional organizations that matter in their chosen field.

1.3 Teaching Students to Use the Code Professionally

How can students use the Code? The Code is a normative code. Historically, there has been a transition away from regulatory codes designed to penalize divergent behavior and internal dissent, toward codes which are more normative, giving general guidance. Such normative codes are only a partial representation of the ethical standards of the profession. [4,5] However, statements in the Joint Code can be used to help students and professionals carefully examine alternative actions when they recognize ethically charged situations.

Because normative codes are incomplete, they should not be considered as mechanical decision procedures to automatically decide what is wrong. The use of normative codes requires moral judgment on the part of the professional. The Codes offer guidance, not an algorithm. But the Code does not leave the reader without support in exercising the required judgment. It includes the following advice about how to make decisions:

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.

Figure 1. The short form of the Code

These Principles should influence software engineers to consider broadly who is affected by their work; to examine if they and their colleagues are treating other human beings with due respect; to consider how the public, if reasonably well informed, would view their decisions; to analyze how the least empowered will be affected by their decisions; and to consider whether their acts would be judged worthy of the ideal professional working as a software engineer. In all these judgments concern for the health, safety and welfare of the public is primary; that is, the "Public Interest" is central to this Code.

The Code establishes a priority in meeting the obligations described in the code. In all decisions the public interest should be the software engineer's primary concern. To reinforce the priority of public well being, the Code explicitly identifies the public good to take priority over loyalty to the employer or profession. Indeed, some of the few serious objections to the Code arose because of this strong position on the public good.

The Joint Code includes these guides to decision-making, but does not include examples that illustrate their use. The next sections of this paper provide three such examples. In the case studies that follow we include a short analysis based on specific clauses in the code, and based on the Code's advice about making professional judgments. Computer science faculty are encouraged to adapt these examples in their classrooms, and to develop new examples based on different case studies. The three cases here were adapted from *Computer Ethics* by Deborah Johnson [6], and are used with permission of the author. The first case was influenced by an earlier paper by Michael C. McFarland. [7]

2. Case Study about Testing: George and the Jet

George Babbage is an experienced software developer working for Acme Software Company. Mr. Babbage is now working on a project for the U.S. Department of Defense, testing the software used in controlling an experimental jet fighter. George is the quality control manager for the software. Early simulation testing revealed that, under certain conditions, instabilities would arise that could cause the plane to crash. The software was patched to eliminate the specific problems uncovered by the tests. After these repairs, the software passed all the simulation tests.

George is not convinced that the software is safe. He is worried that the problems uncovered by the simulation testing were symptomatic of a design flaw that could only be eliminated by an extensive redesign of the software. He is convinced that the patch that was applied to remedy the specific tests in the simulation did not address the underlying problem. But, when George brings his concerns to his superiors, they assure him that the problem has been resolved. They further inform George that any major redesign effort would introduce unacceptable delays, resulting in costly penalties to the company.

There is a great deal of pressure on George to sign off on the system and to allow it to be flight tested. It has even been hinted that, if he persists in delaying the system, he will be fired. What should George do next?

2.1 Particularly relevant clauses in the Joint SE Code

Principle 1. **PUBLIC** Software engineers shall act consistently with the public interest. In particular, software engineers shall, as appropriate:

1.03. Approve software only if they have a well-founded belief that it is safe, meets specifications, passes appropriate tests, and does not diminish quality of life, diminish privacy or harm the environment. The ultimate effect of the work should be to the public good.

1.04. Disclose to appropriate persons or authorities any actual or potential danger to the user, the public, or the environment, that they reasonably believe to be associated with software or related documents.

Principle 3. **PRODUCT** Software engineers shall ensure that their products and related modifications meet the highest professional standards possible. In particular, software engineers shall, as appropriate:

3.10. Ensure adequate testing, debugging, and review of software and related documents on which they work.

Principle 5. **MANAGEMENT** Software engineering managers and leaders shall subscribe to and promote an ethical approach to the management of software development and maintenance. In particular, those managing or leading software engineers shall, as appropriate:

5.01 Ensure good management for any project on which they work, including effective procedures for promotion of quality and reduction of risk.

5.11. Not ask a software engineer to do anything inconsistent with this Code.

2.2 Applying the Code

In this case, Carl Babbage must contend with issues of physical safety that is dependent on software reliability. If we look at this case too narrowly, we might think that the safety of the test pilot is the exclusive safety concern. Although Mr. Babbage does have responsibilities towards the pilot, test pilots know about the risks inherent in their profession, and the test pilot may be quite willing to fly the plane despite Babbage's misgivings. However, the test pilot is not the only one endangered if the software is faulty; anyone *under* the plane is endangered if things go awry. Especially if the test flight might fly over populated areas (and remember that instability might lead the plane in unplanned directions before crashing), many people under the plane are unlikely to have given their consent to "testing" the software. Carl's responsibilities to those people are a vital part of our analysis.

Clause 1.03 makes public safety a priority concern for a software engineer. It is exactly this concern that is central to George's decision. George clearly recognizes this obligation, and the obligation in clause 1.04 to disclose his professional opinion that the software has not been sufficiently certified as safe. Unfortunately, George's superiors have not supported his decision about the software, and are trying to convince him to sign off on the software despite his reservations.

His superiors have put George in a difficult position. Clearly, the Code sections above confirm Mr. Babbage's ethical duty to refuse to sign off on the software before he is reasonably sure of its safety. (We note that for almost all complex software, we can never be entirely sure software is reliable and safe. It is a professional judgment whether or not the software is "safe enough." [8]) By pressuring George to sign off, his superiors are forcing George to choose between his loyalty to his employers (and his continued employment) and his obligation to public safety. As McFarland points out, this is an untenable position. [7] It is hoped that the existence of, and support for, an effective ethics code can help someone in this position; but it is still difficult.

So far our analysis has concentrated on Mr. Babbage and his dilemma. But the Joint Code also requires his managers to act ethically. The clauses in section 5 of the Code prohibit managers from forcing a software engineering employee to violate the code. The Code also makes managers responsible for ensuring that there are processes to ensure the reduction of risks. The managers might object that they have adequate processes, and that the process was followed. Simulation testing revealed problems, and those problems were addressed. The managers are not convinced that Mr. Babbage's suspicions are well founded, and are not willing to jeopardize the project based on his misgivings.

The wording of clause 1.03 in the Code is an important part of our analysis of this case. That clause states that software engineers should approve software only if they have a "*well-founded belief that it is safe*" (our emphasis). The idea of a well-founded belief is key to the dispute between George and his superiors. Perhaps George is right about the software, but perhaps his managers are right. Although the case does not offer many details about George's misgivings, he apparently did not present sufficient evidence to his superiors about the remaining problems in the software. (If the managers were convinced about the seriousness of the remaining problems, it seems unlikely that they would approve a test flight that would likely end in a costly disaster.) Perhaps this dilemma could be resolved to the satisfaction of all parties if the managers agreed to a short term delay not for a major redesign, but for further testing to either confirm George's suspicions, or convince George that the managers are correct, and that the test flight should go on. This resolution would be far better than George signing off on a system he thinks is deficient, and far better than George being fired for not doing so. The standard supported by the Code is to have the burden to demonstrate that the software is safe before deployment instead of having to prove it unsafe before deployment is halted.

3. CASE STUDY ON DATABASE: LEVELS OF SECURITY

Leikessa Jones owns her own consulting business, and has several people working for her. Leikessa is currently designing a database management system for the personnel office of ToyTimeInc., a mid-sized company that makes toys. Leikessa has involved ToyTimeInc management in the design process from the start of the project. It is now time to decide about the kind and degree of security to build into the system.

Leikessa has described several options to the client. The client has decided to opt for the least secure system because the system is going to cost more than was initially planned, and the least secure option is the cheapest security option. Leikessa knows that the database includes sensitive information, such as performance evaluations, medical records, and salaries. With weak security, she fears that enterprising ToyTimeInc employees will be able to easily access this sensitive data. Furthermore, she fears that the system will be an easy target for external hackers. Leikessa feels strongly that the system should be more secure than it would be if the least secure option is selected.

Ms. Jones has tried to explain the risks to ToyTimeInc, but the CEO, the CIO, and the Director of Personnel are all convinced that the cheapest security is what they want. Should Jones refuse to build the system with the least secure option?

3.1 Particularly relevant clauses in the Joint SE Code

Principle 1. **PUBLIC** Software engineers shall act consistently with the public interest. In particular, software engineers shall, as appropriate:

1.01. Accept full responsibility for their own work.

1.03. Approve software only if they have a well-founded belief that it is safe, meets specifications, passes appropriate tests, and does not diminish quality of life, diminish privacy or harm the environment. The ultimate effect of the work should be to the public good.

1.04 Disclose to appropriate persons or authorities any actual or potential danger to the user, the public, or the environment, that they reasonably believe to be associated with software or related documents.

Principle 2. **CLIENT AND EMPLOYER** Software engineers shall act in a manner that is in the best interests of their client and employer, consistent with the public interest. In particular, software engineers shall, as appropriate:

2.05. Keep private any confidential information gained in their professional work, where such confidentiality is consistent with the public interest and consistent with the law.

Principle 3. **PRODUCT** Software engineers shall ensure that their products and related modifications meet the highest professional standards possible. In particular, software engineers shall, as appropriate:

3.01. Strive for high quality, acceptable cost, and a reasonable schedule, ensuring significant tradeoffs are clear to and accepted by the employer and the client, and are available for consideration by the user and the public.

3.03. Identify, define and address ethical, economic, cultural, legal and environmental issues related to work projects.

3.12. Work to develop software and related documents that respect the privacy of those who will be affected by that software.

3.2 Applying the Code

Ms. Jones has competing duties to the people who hired her, the people who work at the company, to her consulting firm (including the people who work for her) and to herself. The Joint Code makes it clear that Ms. Jones must be careful about the issue of privacy; as a steward of sensitive data, she should not lose sight of that responsibility. In our first case, Carl Babbage was most concerned with avoiding physical harm to people; Ms. Jones is concerned with a different kind of harm. Both kinds are important.

At the same time, Ms. Jones needs to balance the need for security with the economic interests of the company that hired her to do this work. Professionals have to make subjective judgments to balance cost and the customer's needs; there cannot be perfect security, and there are never infinite resources. This tension between finite resources and attaining the highest quality policy is a common cause for ethical conflicts.

However, in this case Ms. Jones made a mistake by offering a security "option" to the company that, apparently on later reflection, she thought was inadequate. By not informing the company up front about the necessity and cost for adequate security, she has created a difficult situation, both for ToyTimeInc and for herself. In order to fulfill her obligation to the company employees, she must admit her mistake and remove that insecure system as a viable option, insisting on better security. Although the employees of ToyTimeInc haven't been consulted (at least according to this short description), they clearly will be affected by the decisions ToyTimeInc and Jones make.

One possible objection to Ms. Jones not mentioning the low-security option is that she wouldn't be allowing ToyTimeInc to make an informed decision. But according to the Code, Ms. Jones is responsible for building systems that are beneficial to the public. If the low security system isn't good enough, then she shouldn't pretend that it is. An engineer designing a bridge should not be compelled to include the possibility of building it with shoddy materials in cost estimates.

If the company refuses to upgrade the security, Ms. Jones should probably remove herself from the project if staying in the project will force her to deliver a system she thinks is unethically insecure. (Clause 1.01 is central here.) There are two objections to this suggestion. First, the company will have to find someone else to do the work, and this seems unfair to the company since they were (we assume in good faith) simply agreeing with one of Ms. Jones' suggestions. While this is unfortunate for the company and for Ms. Jones, Ms. Jones' duty to protect sensitive information to a reasonable level of security cannot be brushed aside. A second objection is that if Ms. Jones leaves the project, the company is likely to hire someone else (who perhaps has less ethical scruples) to deliver the job with the unacceptable level of security. Although that may be true, that possible outcome does not absolve Ms. Jones of her responsibility to be an ethical professional. Ms. Jones is first and foremost responsible for her own actions; the next professional hired to take her place will have to wrestle with these responsibilities, but Ms. Jones cannot let that possibility tempt her to dodge her own responsibilities.

There is another effect if Ms. Jones delivers the less secure system. She will have harmed the profession of software engineering by allowing a degradation of the standards for quality software. Such acts will, one software engineer at a time, reduce society's trust in software engineering as a whole.

If ToyTimeInc insists on building the system with inadequate security, Clause 2.05 becomes important. That clause requires Ms. Jones to keep information confidential, *where such confidentiality is consistent with the public interest* (our emphasis). If she thinks the security is sufficiently bad, her obligation to the employees of ToyTimeInc (see clause 1.04) will take priority of the obligation for confidentiality in clause 2.05.

Another possible solution for Jones is for her to tell ToyTimeInc that she and her consulting company will build in better security, but only charge ToyTimeInc for the cheaper option. This will hurt her financially, and may adversely affect her employees. But since Jones made the mistake of offering inadequate security is an "option," she may decide it is best for her professional reputation (and the long term success of her consulting firm) for her to absorb this loss. This option clearly fulfills her obligation to ToyTimeInc employees, although it is less clear that she has been fair to her own

employees who may be harmed by the decision if her company loses money on the ToyTimeInc contract.

4. CASE STUDY ON CONFLICTS OF INTEREST

Juan Rodriguez is a private consultant who advises small businesses about their computer needs. Juan examines a company's operations, evaluates their automation needs, and recommends hardware and software to meet those needs.

Recently, Juan was hired by a small, private hospital interested in upgrading their system for patient records and accounting. The hospital had already solicited proposals for upgrading the system, and hired Juan to evaluate the proposals they'd received.

Juan carefully examined the proposals on the basis of the systems proposed, the experience of the companies that bid, and the costs and benefits of each proposal. He concluded that Tri-Star Systems had proposed the best system for the hospital, and he recommended that the hospital should buy the Tri-Star system. He included a detailed explanation for why he thought the Tri-Star bid was the best.

Juan did *not* reveal to the hospital that he is a silent partner (a co-owner) in Tri-Star Systems. Was Juan's behavior unethical? We will assume for our discussion that Juan evaluated the bids in good faith, and sincerely believed that Tri-Star had given the best bid.

4.1 Particularly relevant clause in the Joint SE Code

Principle 4. **JUDGMENT** Software engineers shall maintain integrity and independence in their professional judgment. In particular, software engineers shall, as appropriate:

4.05. Disclose to all concerned parties those conflicts of interest that cannot reasonably be avoided or escaped.

4.2 Applying the Code

Not all case studies require sophisticated analysis; clause 4.05 clearly labels Mr. Rodriguez's actions as wrong. Mr. Rodriguez did not fulfill his professional obligations when he neglected to disclose his conflict of interest to the hospital. Notice that his sincerity about the superiority of the Tri-Star bid is *not* a central issue here. The central issue is the trust Tri-Star has invested in Juan. If Mr. Rodriguez had disclosed his part ownership in Tri-Star to the hospital, and the hospital had still hired him to evaluate the bids, then Juan could have attempted to do a professional job of evaluation. (Some people might find that difficult, but it is at least theoretically possible.) However, the Code clearly prohibits Juan from taking this job without first giving the hospital the opportunity to judge for itself whether or not they wanted to hire Mr. Rodriguez despite his interest in Tri-Star.

5. CONCLUSIONS

The Joint Code can and should be used in computer science classes. Its use there shows the relevance of professional behavior in the development of quality software. By using the Code and cases, a computer science faculty member can make the study of ethics relevant and compelling to students who are looking forward to careers in computing. The case about Ms. Jones and ToyTimeInc could be used in a database class; the case about Mr. Babbage and the jet fighter could be used during a testing lesson in any programming course; and the case about Mr. Rodriguez and the hospital would be appropriate in a software engineering course.

Students should know that there are standards for ethical, professional behavior. They should realize that professional organizations care about these standards. Computer science faculty can help their students understand the Code and how to apply it in situations the students will soon face in the workplace.

There are many different ways to use ethical case studies in computer science classroom. Three of the ways we've tried in our classrooms are:

A. We have organized an in-class discussion based on a case. Students are given a written copy of the case and the Code, and organized into small groups to answers questions about how to apply the code to the case. Each small group presents its conclusions to the rest of the class at the end of the exercise.

B. We have assigned a case study as homework, including a set of questions for written responses from the students. Alternatively, we have had students write an essay in which the students do their own original analysis instead of answering specific questions.

C. We have staged debates during class. Students are given a written copy of the case and the Code. Then the teacher identifies two possible decisions for the case. (For example, "Fred should do X" and "Fred should do Y.") Then students are invited to debate based on these two possible decisions. One student at a time can defend each position at the front of the class. Students "tag team" to keep the debate going, new students replacing students who have been debating for awhile.

In our classes, we've found that some students are not enthusiastic when asked to study an ethics code. The careful language of ethical codes and the emphasis on professional obligations can be daunting for students. However, most students become more enthusiastic when they use that same ethics code to help them develop and justify a solution to a realistic problem. They find the process of ethical problem solving engaging.

No matter how you introduce the Code in your classes, we think it is useful to at least demonstrate how the Code can be applied to cases. We think it is even more useful to have students practice applying the Code themselves. For more information about teaching techniques for computer ethics, please visit the DOLCE website, established using a National Science Foundation grant, CCLI-DUE-9952841. [9]

6. ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers for their careful reading of the submitted paper. Their suggested changes were all improvements.

7. REFERENCES

[1] D. Gotterbarn, K. Miller, and S. Rogerson. *Software Engineering Code of Ethics and Professional Practice*. (n.d). Retrieved May 28, 2004, from http://csciwww.etsu.edu/gotterbarn/SECEPP .

[2] D. Gotterbarn, K. Miller, and S. Rogerson. Software engineering code of ethics is approved. *Communications of the ACM*, Vol. 42, No. 10 (October 1999), 102-107.

[3] *The Online Ethics Center for Engineering and Science at Case Western University.* Cases. (n.d.). Retrieved March 10, 2004, from http://www.onlineethics.org/eng/cases.html

[4] R. Anderson. The ACM Code of Ethics: History, Process, and Implications. In *Social Issues in Computing*, Huff and Finholt, eds.McGraw Hill, 1995.

[5] D. Gotterbarn. Software Engineering: the new professionalism. In *The Professional Software Engineer*. Colin Myer, ed., 1996.

[6] D. Johnson. *Computer Ethics*, 3rd Edition. Prentice-Hall, 2001.

[7] M. McFarland. Urgency of ethical standards intensifies in computer community. *IEEE Computer* (March 1990), 77-81.

[8] W. R. Collins, K. Miller, B. Spielman, and P. Wherry. How good is good enough? An ethical analysis of software construction and use. *Communications of the ACM*, Vol. 37, No. 1 (January 1994), 81-91.

[9] *Developing On/Off Line Computer Ethics*. (n.d.). Retrieved May 28, 2004 from http://csethics.uis.edu/dolce/.

View publication stats