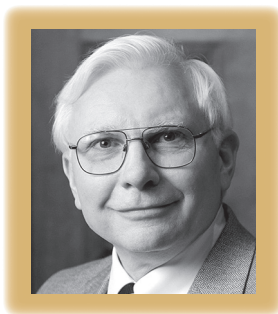
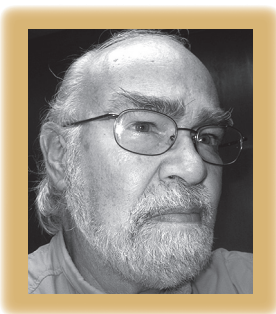


Unmasking Your Software's Ethical Risks

Don Gotterbarn and Keith Miller

It's difficult to fully address all our professional obligations as software engineers. Our training focuses on avoiding technical failures, but unfortunately our systems sometimes have unintended consequences. We need to develop products to avoid unintended negative impacts on society, people, and the environment.

Professional responsibility requires that we identify the morally salient features of a situation.



Some issues are relatively easy to spot; for example, we shouldn't lie to clients, we shouldn't bribe inspectors, and we should respect people's privacy. But some ethical and social risks are harder to recognize. Even developers with the best intentions have walked into ethical traps.

In a previous column, Duncan Hall asks, "How might professional software engineers contribute to society at large? They can develop awareness of society's wider issues and how individual decision-making can influence outcomes, both favorably and unfavorably."¹ We wish in this article to address Hall's question and explore what to look for during a software development project.

When we study technical problems, we apply the project's constraints and priorities to find acceptable possible solutions and choose among them. Here are four suggestions for considering ethical constraints during that process.

Look for human values in technical decisions

Look for Human Values in Technical Decisions

It should become routine for software engineers to look for the human values inherent in technical

decisions. Think early and often about the people your systems affect—both those directly related to the development, clients and users, but also those affected by that use. In addition to pilots and airlines, passengers and people living under flight paths are important stakeholders in the quality of avionics software. Think broadly, and include considerations such as the environment, culture, and social impacts.

Software engineers must be aware of legal constraints and obligations, but ethics calls us to go further. How do our creations support or harm core values in both the short and long term? Values to maximize include ability, security, knowledge, freedom, opportunities, and resources.

Be conscious of the context of your deployed system to anticipate potential trouble spots. Project costs and system function are very important, but there are other risks whose identification requires explicit examination of the context of the system. Ask about the special circumstances generated by the nature of the project.

Identify Who Will Be Affected

Three questions can serve as a simple heuristic to help you identify stakeholders and affected parties. Whose behavior or work process, whose circumstance or job, and whose experiences will be affected by the development and delivery of this system?

You can answer these questions thoroughly by using stories or scenarios about how the software will be used, perhaps in different contexts. Playing out these scenarios can help you uncover problems. An XP version of Windows Accessibility Wizard has a function to enable the visually challenged to increase the font size. After the user selects the font they can read (for example, 16-point text), the instructions on how to apply this selection appear in another window in an 8-point font. Is this a technical mistake? Surely. But it has real consequences

for a class of users, and that has ethical significance.

The relationship between stakeholder identification and understanding the context of the situation is iterative. A better understanding of the context leads to identification of previously unidentified stakeholder groups, and the identification of other stakeholders' involvement leads to a better understanding of the context. The combination of studying stakeholders and context, using our professional judgment guided as by the Code of Ethics, enables us to identify the system's potentially adverse effects.

Examine How Stakeholders' Rights and Obligations Will Be Affected

Look for responsibilities and relationships that the system will reinforce or disrupt. Each stakeholder has certain rights and obligations with respect to the other stakeholders. The system's design, development, and deployment may facilitate some of those rights and obligations and hinder others. Ask yourself these questions:

- Is the system likely to improve some relationships (make them more just, less hostile, more efficient)?
- Are there any ways in which the system is likely to degrade some relationships (make them less just, more hostile, less efficient)?
- If you were one of the people most affected, would you regard the system as a help, a hindrance, or neutral?

Sometimes the focus on technical issues impinges on our sensitivity to critical subtleties. For example, some automated weapons systems determine targets and aiming information. Using a highly complex management-by-exception system, such systems will fire on the selected target unless a human controller overrides it in a required period of time. According to Mary Cummings, the fact that operators were given only 10 seconds to veto a computer's firing solution is suspected as contributing to the downing of a British and an American plane.² Clearly a terrible tension exists here in the design of such systems: if the time-to-veto is too long, enemy targets could escape, perhaps resulting in many casualties; if the time-to-veto

is too short, then a false positive could result in friendly targets being shot down. The gravity of this situation illustrates that setting the number of seconds to delay is a life-and-death technical decision. This decision is crucial to the software design, to the training of the people who must make the difficult go-or-no-go decisions, and to the rules of engagement set by policy-makers. The difficult balancing of trade-offs requires us to exercise wisdom and care, both of which require time for careful consideration.

Review Relevant Professional Standards to Help Identify Issues

Does the IEEE/ACM Software Engineering Code of Ethics and Professional Practice³ include advice relevant to your project?

The Code (www.computer.org/ethics) was developed by a joint IEEE-CS/ACM task force and has since been adopted by other professional societies and corporations as a model of ethical software engineering practice. The following excerpts include tips about how to spot ethical problems and avoid ethical traps.


The Code's preamble provides a strategy for thinking about these issues:

- "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" Does your system let people determine their own goals and act freely consistent with human well-being? Does it respect autonomy?
- "To consider how the public, if reasonably well informed, would view their decisions" Does people's acceptance of the system you developed depend on them having a limited understanding of technology?
- "To analyze how the least empowered will be affected by their decisions" Is your system useful only to those at a particular intellectual or physical level? Is 10 seconds the right time limit for a life-critical decision?
- "And to consider whether their acts would be judged worthy of the ideal professional working as a software engineer." Is your approach consis-

tent with the best standards of software engineering?

If you're uncomfortable with a system or wonder if a problem exists, it's best to assume there is a problem and review the Code with your question in mind. Several clauses may strike you as relevant, and some will be of more consequence than others. To resolve any conflict among principles, one principle trumps all others: "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 is the bottom line. Software engineering is a service profession. What we develop must be designed for the public good.

As in any software development process, answering initial questions often reveals new, critical questions. The same is true when identifying ethical issues. As in all good engineering, the kinds of issues identified in one case can be useful in identifying others.

What we've described here is how to *recognize* ethical issues. Sometimes, once you spot an ethical problem, resolving it is straightforward. At other times, it might require extensive deliberation and negotiation to work through. But one thing is certain: unless you recognize the problem, you won't be able to manage it. And unmanaged ethical issues have a nasty habit of causing serious problems down the road. 

References

1. D. Hall, "The Ethical Software Engineer," *IEEE Software*, July/Aug. 2009, pp. 9–10.
2. M. Cummings, "Automation and Accountability in Decision Support System Interface Design," web.mit.edu/aeroastro/labs/halab/papers/Cummings_JTS.pdf.
3. D. Gotterbarn, K. Miller, and S. Rogerson, "Software Engineering Code of Ethics and Professional Practice," *Computer*, Oct. 1999, pp. 84–88; www.computer.org/ethics.

Donald Gotterbarn is professor emeritus in the Department of Computer Science at East Tennessee State University. He chairs the ACM's Committee on Professional Ethics and chaired the executive committee that developed the Software Engineering Code of Ethics and Professional Practice. Contact him at gotterbarn@comcast.net.

Keith W. Miller was recently awarded the Louise Hartman Schewe and Karl Schewe professorship in the Department of Computer Science at the University of Illinois at Springfield. Contact him at miller.keith@uis.edu.