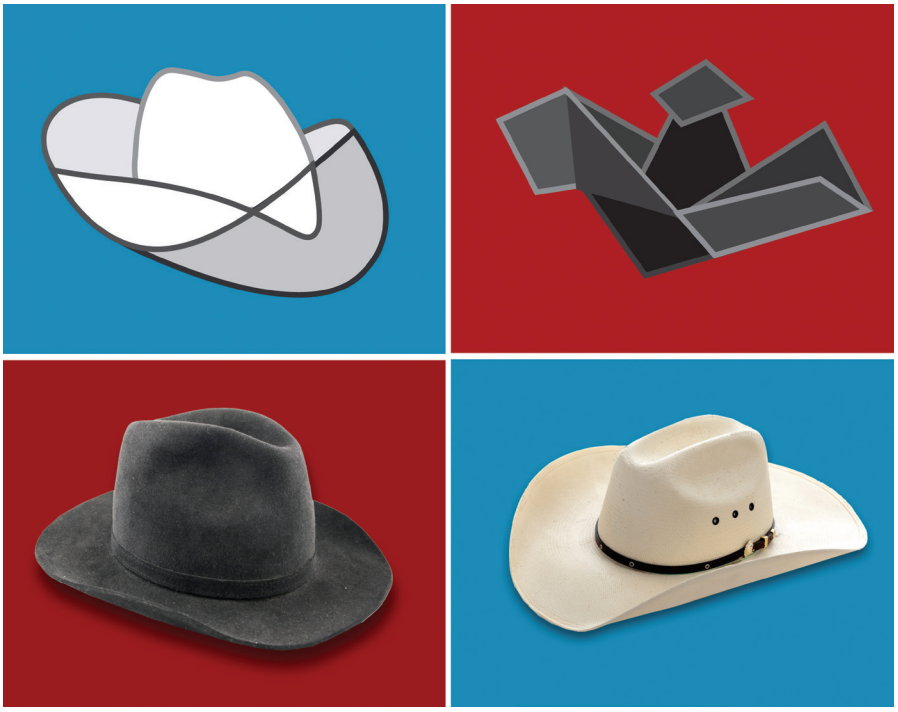
DOI:10.1145/2618103

Chuck Huff and Almut Furchert

Computing Ethics
Toward a Pedagogy of Ethical Practice

Teaching computing ethics in a manner that allows students to address both abstract ethical knowledge and actual ethical practice.



There is a tension at the heart of our curriculum in computer ethics. We want to prepare students for the wide range of ethical concerns in the practice they will find upon graduation. But primarily, we teach knowledge, intermixed with such practice as fits in the spaces of the lecture/discussion course.

Our recent work on "moral exemplars" in computing[a] found a variety of ways that professionals approached doing good.[7] Some designed systems to help individuals or organizations. Alan Newell, for instance, has been designing software and hardware to help the disabled—his lab did some of the earliest work on predictive spelling systems. Others focused on changing social systems, like Stephanie (Steve) Shirley who brought many women into computing through her pioneering efforts in software consulting.

This variety of ethical practice is a normal finding in work on moral exemplars.[2] But among the variety, we also find similarity: the exemplars' language in the narrative interviews we conducted was filled with expressions implying significant ethical commitments (for example, "to look for the needs that people and organizations have;" "quality, fitness for purpose;" "openness, transparency"). These fit items one can find in the software engineering ethics code (see the accompanying table). Oddly, although some helped write ethics codes, *none of the interviewees ever mentioned any code of ethics*, even when asked specifically about principles.

So on what were they basing their moral action? Among those who were designing computing systems, the center of their craft was recognizing the needs of stakeholders and using their expertise to reframe those needs into things that computing could help them do. They used social expertise to listen to individuals and organize networks of individuals to get things done. This blended into technical expertise in socio-technical and requirements analysis. And it included deep technical skill in encryption, database design, and networking protocols, among other topics. Thus, our exemplars were exercising an expertise or skill to integrate their ethical commitments with their knowledge and

a We did in-depth interviews with a careful sample of 24 moral exemplars in computing, in the hope that understanding the lives of these experts in ethical computing practice we might learn how to teach that expertise. We are grateful to the exemplars who took time for our project, and to those whose names we use by permission.

know-how of technical and social systems.

Aristotle's early analysis of ethical action hints that ethical knowledge (the what) and ethical skill (the how) are intermingled in any practice. Alongside knowledge of the good, we need to gain practical wisdom (phronesis) that guides our ethical action. We acquire practical wisdom by practicing, we "become builders by building and lyre players by playing the lyre; so too we become just by doing just acts..."[1] Here lies what the German philosopher Hans-Georg Gadamer[5] has described in the context of medical practice as the application problem. The unavoidable difficulty for any medical professional is "applying [medical] knowledge in the concrete case." Gadamer also claims this application problem is "irreducible ... where ever knowledge in general needs to be applied." Bridging this gap is an act of constant evaluation, reasoning, and decision making to translate what we know into doing—and this translation unavoidably brings the ethical dimension into every aspect of professional practice. If we want to teach ethical practice in computing, we must therefore ask what competencies, skills, or virtues are needed to translate ethical knowledge into ethical practice in the field.

Herein lies an uncomfortable tension: While the ethics code is full of the obligation to design systems with ethical concern in mind, the typical computing ethics textbook does not help one learn how to do that. One can learn knowledge about legal, philosophical, and societal issues of privacy in a classroom, and even practice thinking about these issues with cases. Some of these skills (identifying stakeholders, making an ethical argument) are useful when one has the problem of designing a computing system, but one must remember them, and remember they apply, and know how to adapt them to the concrete case. The *Social Issues and Professional Practice* ACM curriculum[3] points us in the direction of balancing knowledge with practice by setting "usage" objectives (what one must be able to do with the knowledge).

But to implement these recommendations requires a pedagogy of ethical practice. How does one, for instance, "... address ethical ... issues related to work projects" per section 3.03 of the software engineering ethics code? It requires knowledge about ethical issues in general, but also know-how to identify them in a particular project. It requires knowledge about the socio-technical system, but also know-how about how to acquire such knowledge. It requires knowledge about best practices, but also know-how about selecting and adapting those practices for a specific social and organizational context.

Research suggests one best acquires expertise by long practice, informed by knowledge and theory of the domain and with coached feedback about performance that is immediate and explicit.[4] This fits with what we have heard from the philosophical approaches and the narratives told by our moral exemplars. All three lead us to think our moral exemplars have developed an expertise in the ethical practice of their profession; they have extensive, skilled practice guided by ethical commitments and knowledge.[b,6] This is perhaps why our ethical experts do not cite the code. They learned their ethical design skills by practicing them until they became automatic, thoughtful, goal-directed action. They did not reference the high-level principles of the code since they were integrated into their skilled practice and had become part of their expertise.

We have developed a computing ethics class that attempts to teach ethical expertise in practice. Students provide clients with consulting on ethical issues associated with their computing systems (see http://pages.stolaf.edu/csci-263-2014/). They construct a model of the socio-technical system based on interviews with the client and then look for social and ethical issues (including safety, privacy, property, justice) at the individual to the societal level.[8] They scope their project based on time, resources, and urgency of the issues. They then analyze those issues in that socio-technical system using human-computer interaction (HCI) approaches to data

b Moral expertise is a central aspect of a forthcoming book titled *Taking Moral Action*.

Items from the ACM/IEEE Software Engineering Code of Ethics.

These 16 items are selected because they are clearly relevant to having competence in the consideration of ethical aspects in the design of software.

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.

1.07. Consider issues of physical disabilities, allocation of resources, economic disadvantage, and other factors that can diminish access to the benefits of software.

2.07. Identify, document, and report significant issues of social concern, of which they are aware, in software or related documents, to the employer or the client.

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.

3.02. Ensure proper and achievable goals and objectives for any project on which they work or propose.

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

3.04. Ensure that they are qualified for any project on which they work or propose to work by an appropriate combination of education and training, and experience.

3.05. Ensure an appropriate method is used for any project on which they work or propose to work.

3.06. Work to follow professional standards, when available, that are most appropriate for the task at hand, departing from these only when ethically or technically justified.

3.08. Ensure that specifications for software on which they work have been well documented, satisfy the users' requirements, and have the appropriate approvals.

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

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

3.13. Be careful to use only accurate data derived by ethical and lawful means, and use it only in ways properly authorized.

3.14. Maintain the integrity of data, being sensitive to outdated or flawed occurrences.

4.01. Temper all technical judgments by the need to support and maintain human values.

These 16 statements represent 20% of the total of 80 statements in entire code, and over 70% of the items in the section on responsibilities having to do with the production of a product (section 3).

collection (for example, interviewing, focus groups, active and passive observation, think-aloud protocols and so forth). Finally, they construct proposed solutions and make design recommendations. The course is built on the dialogue among concern for ethical issues, knowledge about computers in context, and the practice of working for a client. They learn about ethical and social issues from a textbook[9] and from constructing solutions for their clients. They learn about socio-technical systems from the ethics text and also by using methods from an HCI text.[10] During the past decade, teams from the course have produced over 50 studies for non-profit, for-profit, and internal clients. They have made recommendations for Web design, for customizable privacy settings in social networks, for medical database systems, key-card access systems, productivity software for robotic manufacturing, cloud-based systems, and many more. The goal is that students leave the course having learned how to hold together their ethical concern, knowledge, and practice. This is the beginning of expertise in ethical issues in computing design.

There already exist a variety of methods for the incorporation of ethical concerns in systems design that might be used in courses (see Shilton[11] for an overview). We hope our suggestions here will encourage further experimentation to incorporate them into the curriculum. For instance, in those programs that have HCI labs, one might arrange for significant overlap between the lab and a supporting course on ethical issues. Programs with project courses in other areas (such as networking, information management, parallelism) could incorporate into those projects some of the ethical design approaches that might be taught in more detail in a computer ethics course.

As educators of those who will design future technology, we have a responsibility to prepare our students to practice their craft in a way that integrates their ethical concern into their work. We propose that it is necessary and possible to teach computing ethics know-how that helps students to navigate between abstract ethical knowledge and its actual ethical practice. By doing so, the students gain experience and expertise in applying what they know in the concrete case. C

References

1. Aristotle. Nicomachean ethics. In *The Basic Works of Aristotle*, R. McKeon, Ed. Random House, New York, 1941, 927–1112.
2. Colby, A. and Damon, W. Some do care: Contemporary lives of moral commitment. Free Press, New York, 1992.
3. *Computer Science Curricula 2013.* ACM/IEEE Computer Society, 2013.
4. Dreyfus, H.L. and Dreyfus, S.E. The ethical implications of the five-stage skill-acquisition model. *Bulletin of Science, Technology, & Society 24* (2004), 251–264.
5. Gadamer, H.-G. *The Enigma of Health.* Stanford University Press, Stanford, CA, 1996.
6. Huff, C.W. From meaning well to doing well: Ethical expertise in the GIS domain. *Journal of Geography in Higher Education*, in press.
7. Huff, C.W. and Barnard, L.K. Good computing: Moral exemplars in the computing profession. *IEEE Technology and Society Magazine* (2009), 47–54.
8. Huff, C.W. and Martin, C.D. Computing consequences: A framework for teaching ethical computing. *Commun. ACM 38*, 12 (Dec. 1995), 75–84.
9. Johnson, D. *Computer Ethics, 4th Edition.* Pearson, New York, 2009.
10. Rogers, Y., Sharp, H., and Preece, H. *Interaction Design: Beyond Human-Computer Interaction.* Wiley, New York, 2011.
11. Shilton, K. This is an intervention: Foregrounding and operationalizing ethics during technology design. In *Emerging Pervasive Information and Communication Technologies (PICT): Ethical Challenges, Opportunities, and Safeguards, 11*, K.D. Pimple, Ed., Springer, New York, 2013, 177–192.

Chuck Huff (huff@stolaf.edu) is a professor of psychology at St. Olaf College, Northfield, MN.

Almut Furchert (furchert@stolaf.edu) is a senior visiting research fellow in the Hong Kierkegaard Library at St. Olaf College in Northfield, MN.

This work was partially supported by NSF grants DUE-9980786, DUE-9972280, SES-0217298, and SVS-0822640. We are grateful to Don Gotterbarn, Keith Miller, and Herman Tavani for conversations and advice about this column.