SLICKROCK II

Value-Sensitive Design

Batya Friedman *Colby College and The Mina Institute*

Values emerge from the tools that we build and how we choose to use them. Yet, in most of the current practice in designing computer technology and the related infrastructure of cyberspace, little is said about values.

After all, values—especially moral values—can be controversial. They can also seemingly conflict with economic goals and can be difficult to articulate clearly and to translate into meaningful processes and designs. What then is there to be said about accounting for human values in system design? How should we say it?

Over the past decade, my colleagues and I have addressed these questions as we have taken an active stance toward creating computer technologies that—from an ethical position—we can and want to live with [2, 4 – 7]. I have also sought to develop a framework for understanding how specific values play out in the design of computer systems, including how these values can be undermined or promoted by the technology. In this article, I discuss projects (conducted in collaboration with Helen Nissenbaum) that concern two values: user autonomy and freedom from bias. I conclude with some reflections on the importance of value-sensitive design, justifiable limitations on its implementation, and ways in which it complements economic mandates.

User Autonomy[1]

Consider a recent workstation design that comes from a leading computer hardware and networking company in the United States (see [20] for a detailed discussion). The workstation was designed to support speech input and multimedia, and thus included a built-in microphone. Nothing strange here. Except that the microphone automatically recorded audio information whenever the workstation was on. Now, imagine that you are in the middle of a video-conferencing session and a visitor comes into your office, or the phone rings. The only way to ensure audio privacy is to turn off the application, which is a cumbersome solution. Alternatively, a simple solution existed in the design process (ultimately vetoed by the design team): to install a hardware on/off switch on the microphone at the cost of 25 cents.

This example illustrates how hardware design can either hinder or help the user's ability to control the technology. More generally, it speaks to the value of user autonomy. By this term we refer to individuals who are self-

[1]User autonomy was the topic of a workshop Nissenbaum and I organized at CHI'96 [6]. Participants in the workshop included Bay-Wei Chang, Ise Henin, David Kirsh, Pekka Lehtio, Nicole Parrot, and Mark Rosenstein, They contributed to the ideas developed here.

Batya Friedman,
Department of
Mathematics and
Computer Science,
Colby College,
Waterville, ME 04901.
E-mail:
b_friedm@colby.edu.

determining, who are able to decide, plan, and act in ways that they believe will help them to achieve their goals and promote their values. People value autonomy because it is fundamental to human flourishing and self-development [8, 9].

How can designs promote user autonomy? From the previous example, there might seem to be a simple answer: If autonomous individuals need to have freedom to choose means and ends, then it could be said that whenever possible and at all levels, designers should provide users the greatest possible control over computing power. On closer scrutiny, however, there is a more complex picture. After all, think of a text editor that helps a non-technically minded user to create consistent and well-formatted documents. Such users will have little interest in explicitly controlling lower levels of operation of the editor even though they will appreciate control over higher level functions. They will have little interest, say, in controlling how the editor executes a search and replace operation or embeds formatting commands in the document, and more interest in controlling the efficient and effective formatting of the document. In this case, achieving the higher order desires and goals, such as efficiently producing a good-looking document, will enhance autonomy, whereas excessive control over all levels of operation of the editor may actually interfere with user autonomy by obstructing users' ability to achieve desired goals.

In other words, autonomy is protected when users are given control over the right things at the right time. Of course, the hard work of design is to decide these whats and whens. Accordingly, Nissenbaum and I have begun to identify aspects of systems that can promote or undermine user autonomy. The four that I discuss here are system capability, system complexity, misrepresentation of the system, and system fluidity.

System Capability

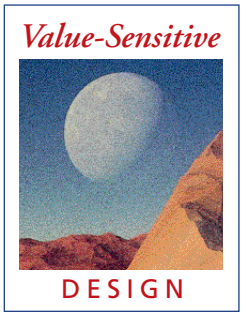
Recall the introductory example of a workstation microphone with no hardware on/off switch. Here (assuming no software fixes), users lack the capability to easily interrupt a videoconference for a private office conversation. Similarly, users sometimes need but are denied access to low-level manipulations from within an operating system. Or consider a state-of-the-art 3-D rendering system designed by a leading software company to support artists with the animation process. The system designers assumed users would work from hand-drawn or computer-generated sketches. But when some users preferred to work from video images, it was discovered that there was no way to connect video input to the system. All three cases illustrate that user autonomy can be undermined when the computer system does not provide the user with the necessary technological capability to realize his or her goals.

System Complexity

In some instances, systems may supply users with the necessary capability to realize their goals, but such realization becomes difficult because of complexity. We can see this problem clearly with the recent proliferation of features in many word processing programs. Although more features mean more capability, more features often increase a program's complexity and thereby decrease its usability, especially for the novice user. Granted, a designer can rightfully expect users to spend time learning. But the question is How much time? Moreover, other problems of system complexity arise from a mismatch between the abilities of the user—for example skill level, memory, attention span, computational ability, and physical ability—and those required to use the system efficiently.

Misrepresentation of the System

Users can experience a loss of autonomy when provided with false or inaccurate information about the computer system. Imagine, for example, the package copy for a news filter that states "this news filter is as good as the best personal assistant." Given the state of the field, such hyperbole will mislead a user who believes the package copy and, thus, develops inaccurate expectations of the software agent's ability to realize the user's goals. Or consider the following vignette related by a colleague in her remarks in a panel at CHI 95. My colleague had recently visited a MOO—one that



users visit with the mutually understood goal of meeting and interacting with others. While there, she "walked" into a bar and started to chat with the bartender. Several minutes into the conversation, she had a realization: the bartender was a bot—not a person taking part in on-line conversation! So much for thinking she had met a person she might like to know better. There is also the loss of time, the feeling of deception, and the residual doubt "will the next encounter seemingly with a person actually be with someone's code?" Such experiences can undermine the quality of interactions in an electronic community and ultimately affect the user's original goals for participation.

System Fluidity

With the proliferation of electronic information, automated filters—such as mail agents and news filtering software—have become increasingly common. Now, let's assume that initially a filter does a good job of discarding or never even retrieving unwanted information. Over time, however, the user's goals can change. A single woman, for example, might have little interest in company e-mail on maternity leave and child care, but 3 years later and expecting a baby, the same messages might be wanted. But because the filter does not provide the woman with information about what was previously discarded, she has no way of assessing what she is currently missing. Indeed, she would have no hint that the mail agent even needs to be reprogrammed or retrained. The point here is that users' goals often change over time. Thus to support user autonomy, systems need to take such change into account and provide ready mechanisms for users to review and fine-tune their systems [11, 12].

Bias in Computer Systems[2]

In its most general sense, "bias" means simply "slant." Given this undifferentiated usage, bias can describe both moral and nonmoral circumstances. In our work, however, Nissenbaum and I have been focusing on computer technologies with biases as a source of moral concern; thus, we use the term bias in a more restricted sense. We say that a computer technology is biased if it systematically and unfairly discriminates against certain individuals or groups of individuals in favor of others. A technology discriminates unfairly if it denies an opportunity or a good, or if it assigns an undesirable outcome to an individual or group of individuals on grounds that are unreasonable or inappropriate.

How then does bias become embedded in computer systems? In our study of 17 existing systems, we have identified three overarching ways: preexisting bias, technical bias, and emergent bias.

Several minutes into the conversation, she had a realization: the bartender was a bot—not a person taking part in on-line conversation

Preexisting Bias

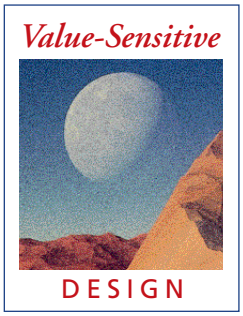
Preexisting bias has its roots in social institutions, practices, and attitudes. When computer technologies embody biases that exist independently of, and usually before, the creation of the technology, we say that the technology embodies preexisting bias. Preexisting biases may originate in society at large, in subcultures, or in formal or informal organizations and institutions. They can also reflect the personal biases of individuals who have significant input into the design of the technology, such as the client or the system designer. This type of bias can enter a technology either through the explicit and conscious efforts of individuals or institutions, or implicitly and unconsciously, even despite the best of intentions. Consider, for example, software that a colleague purchased for his school-

[2]Much of the material discussed here has been adapted from several sources [3, 5].

age daughter. He writes, "Well, of course, the first thing that happens is this: you get to choose which one of three basic adventurers you want to be—a male thief, a male magician, or a male warrior. Nice choice for a young girl, huh?" [3], p. 49]

More formally, Huff & Cooper [10] conducted a study showing that software designers sometimes unknowingly design software that is more aligned with males than with females. In the study, subjects were asked to propose designs for software to teach seventh graders the correct use of commas. One group of subjects was asked to design the software for seventh-grade boys, the second group to design for seventh-grade girls, and the third group to design for seventh-graders, gender unspecified. Huff and Cooper reported that along a number of dimensions the designs proposed by subjects in the gender-unspecified group closely resembled the designs proposed by subjects who designed for boys and were significantly different from the designs proposed by subjects who designed for girls. The study illustrates how preexisting biases, in the form of expectations about which software will appeal to each gender, coupled with the implicit assumption that the generic user of software is likely to be male, can influence design and give rise to bias in software.

Whereas gender bias is deeply embedded in Western society, other biases can serve individual or corporate interests. The Sabre and Apollo computerized airline reservation systems that have been charged with intentional bias [17] are one example. The limited size of display screens often means that all possible flights that match a traveler's request cannot be shown simultaneously on the screen. The flights that are shown on the initial screen (or the part of the file that is visible before the user scrolls) are more likely to be selected by travel agents and their clients [19]. In the Sabre and Apollo systems, because of details in the algorithm certain airlines regularly appear on the first screen and thus have systematic advantages over those airlines whose flights do not.



Technical Bias

In contrast to preexisting bias, technical bias arises from the resolution of issues in the technical design. Sources of technical bias can be found in several aspects of the design process, including

- Limitations of computer tools such as hardware, software, and peripherals;
- The process of ascribing social meaning to algorithms developed out of context;
- Imperfections in pseudo-random number generation; and
- The attempt to make human constructs amenable to computers—when, for example, we quantify the qualitative, make discrete the continuous, or formalize the nonformal.

The use of computer punch card tallying systems for national and local elections provides a case in point. Undereducated groups are more likely to not understand how the computerized system works and, thereby, to invalidate their own votes by either not voting for a position, or by voting for more than one person per position [1]. When this occurs, features of the interface create unfair difficulties for undereducated voters and thus pose serious problems for fair elections (see also [16]).

Graphical user interfaces (GUIs) provide another example of technical bias. Before the invention of GUIs, many visually impaired individuals were active members of the computing community and many supported themselves as computer professionals. GUIs then appeared and rapidly became a standard. Key to our discussion, the development of GUIs was motivated not by preexisting social bias against visually impaired computer users, but rather by new technical developments in graphics-oriented hardware and displays coupled with the belief in the old adage "a picture is worth a thousand words." Although the new GUI standard improves access to the technology for many, it has effectively shut out visually impaired computer users. On a more positive note, to counter this technical bias the computer industry has invested substantially in attempting to design technical means that would allow visually impaired users access to the information buried in the graphical user interface.

Emergent Bias

Although it is almost always possible to identify preexisting bias and technical bias in a design at the time of creation or implementation, emergent bias arises only in a context of use by real users. This bias typically emerges some time after a design is completed, as a result of a change in societal knowledge, user population, or cultural values. For example, much of the educational software developed in the United States embeds learning activities in a game environment that rewards competitive and individual playing strategies. When such software is used by students with a cultural background that eschews competition and instead promotes cooperative endeavors, such students can be placed at a disadvantage in the learning process. Or consider the National Resident Match Program (NRMP), a computerized matching system for assigning medical residents to hospital residency programs, that was designed and implemented in the 1950s. The initial algorithm used by the NRMP placed couples (where both members of the couple were residents seeking a match) at a disadvantage in the matching process compared with their single peers. However, this bias emerged only in the late 1970s and early 1980s when increasing numbers of women entered medical school and a growing number of couples sought residencies [14, 15]. To the credit of those who oversee the NRMP, recent revisions in the algorithm place couples more equitably in matches.

Well, of course, the first thing that happens is this: you get to choose which one of three basic adventurers you want to be—a male thief, a male magician, or a male warrior. Nice choice for a young girl, huh?

Design Methods to Minimize Bias

As the computing community develops a better understanding of bias in system design, we can correspondingly develop techniques to avoid or minimize it. Certainly an initial step involves identifying or "diagnosing" bias, and to do so during the earliest stages of the design phase, when negotiating the system's specifications with the client and constructing the first prototypes. Then comes the task of remedying. Some current designers, for instance, address the problem of handedness by allowing the user to toggle between a right-handed or a left-handed configuration for user input and screen display. Elsewhere, members of the Archimedes Project at Stanford University are developing an approach to designing for people with physical disabilities [13]. More broadly, the HCI community has considerable experience in designing usable systems for both color-blind and color-sighted individuals by the simple means of redundant information: Whatever pertinent information is communicated through color is also communicated in some other form. The more success we have with specific designs, the more readily we can develop more systematic approaches to both process and technique. For example, in their attempt to systematize attention to bias in design, NYNEX has begun to embed methods for minimizing bias into their certified design processes for ISO 9001 registration [21]. Eventually, the standards themselves should require such value-sensitive processes.

Conclusion

Although computer technology is expensive to develop, it is comparatively inexpensive to produce and disseminate, and thus the values embedded in any given implementation are likely to be widespread, pervasive, and systematic. Moreover, unlike with people with whom we can disagree about values, we cannot easily negotiate with the technology. Although inattention to moral values in any enterprise is

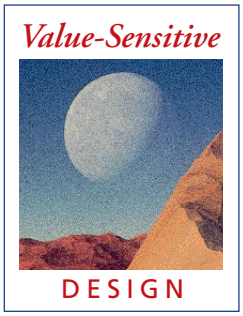
disturbing, it is particularly so in the design of computer technology. Thus, I have suggested that in the design of computer technology we value human values as understood from an ethical standpoint.

This is not to say that one or more such values, such as autonomy or freedom from bias, necessarily overrides others. For example, when systems are employed in contexts that seriously affect human welfare, such as in an air traffic control system, we may need to restrict autonomy to protect against a user with malicious intentions or well-intentioned users guided by poor judgment. Neither is it always clear how to balance competing values. Take, for example, the issue of standardization in design. On the one hand, some forms of standardization will restrict how users can control technology. On the other hand, standardization can free users from the burden of relearning how to work with the technology when they switch among stations or systems. In these situations, standardization provides users with not lesser but greater control over the technology.

Certainly, more thinking is needed on how to balance human values. For example, it may be that reconciling the ideals of standardization and autonomy depends on identifying the appropriate level of user control. It will also be necessary to consider the relationships between moral values and economic goals. Part of the difficulty here is that economic goals are themselves human values, and at times even moral ones, as when people strive through economic means to attain autonomy, that is, the freedom and responsibility to attain necessary goods for themselves and their families. But personal and corporate economic goals also have a long history of undermining moral values: when, for example, a drive for profits runs roughshod over the development of safe products and the fair treatment of workers and customers. In such cases, it should go without saying that the economic needs to give way to the moral.

Moral values can also support economic goals. Yet this point often gets lost in bottom-line cost–benefit analyses. For example, protecting user autonomy means giving users control at the appropriate level over their machines. This protection can translate into marketable features, such as of privacy and security. Minimizing bias in a design also likely leads to a larger market share because such systems are typically accessible to a greater diversity of users (e.g., users who are color-sighted or color-blind, right-handed or left-handed, male or female). Moreover, even when a greater market share is not directly anticipated there is the goodwill of customers who associate a company or its product with moral purposes. Such an association is difficult to quantify economically, no doubt; but so too are the economic benefits of advertising. It is also worth noting that retrofitting a design is vastly more costly than building things right the first time.

Which brings us to what it means to build things "right." Right by what criteria? Some currently accepted ones in system design include reliability, efficiency, and correctness. Yet there is a growing consensus that we need also to include criteria that embody or at least help foster core human values [18, 21]. Thus, in future work we need to continue to conceptualize values carefully and then study them in both laboratory settings and organizations. Moreover, we will need to examine our own design practices from this perspective. By such means, designs could be judged poor and designers negligent. As with the traditional criteria of reliability, efficiency, and correctness, we do not require perfection in value-sensitive design, but a commitment. And progress.



About the Author

Batya Friedman is Associate Professor of Computer Science at Colby College. She received both her B.A. and her Ph.D. from the University of California at Berkeley. Her areas of specialization are human-computer interaction and the human relationship to technology. She has written numerous research articles in addition to designing educational software and consulting on human values in system design. Currently she is editing a book titles *Human Values and the Design of Computer Technology* to be published shortly by the Center for the Study of Language and Information, Stanford University.

References

1. Dagger, R. . Annals of democracy. The New Yorker (November 7, 1988): 40–46, 51–52, 54, 56, 61–68, 97–100, 102–108.

2. Friedman, B. (ed.). Human values and the design of computer technology. Center for the Study of Language and Information, Stanford University, Stanford, Calif., in press.

3. Friedman, B., Brok, E., Roth, S. K., and Thomas, J. Minimizing bias in computer systems: CHI '95 Workshop. SIGCHI Bulletin 28, 1 (1996): 48–51.

4. Friedman, B. and Kahn, P. H., Jr. Human agency and responsible computing: Implications for computer system design. Journal of Systems Software 17 (1992): 7–14.

5. Friedman, B. and Nissenbaum, H. Bias in computer systems. ACM Transactions on Information Systems 14, 3 (1996): 1–18.

6. Friedman, B. and Nissenbaum, H. User autonomy: Who should control what and when? Conference Companion of the Conference on Human Factors in Computing Systems, CHI '96 Association for Computing Machinery, New York, April 1996, p. 433.

7. Friedman, B. and Winograd, T. (eds.). Computing and social responsibility: A collection of course syllabi. Computer Professionals for Social Responsibility, Palo Alto, Calif., 1990.

8. Gewirth, A. Reason and morality. University of Chicago Press, Chicago, 1978.

9. Hill, T. E., Jr. Autonomy and self-respect. Cambridge University Press, United Kingdom, 1991.

10. Huff, C. and Cooper, J. Sex bias in educational software: The effect of designers' stereotypes on the software they design. Journal of Applied Social Psychology 17 (1987): 519–532.

11. Laurel, B. Interface agents: Metaphors with character. In B. Laurel (ed.), The art of human-computer interface design Addison-Wesley, Reading, Mass., 1990, pp. 355–365.

12. Malone, T. W., Lai, K. Y., and Fry, C. Experiments with Oval: A radically tailorable tool for cooperative work. Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW '92), Toronto, Ontario, November 1992.

13. Perry, J., Macken, E., Scott, N., and McKinley. Disability, inability and cyberspace. In B. Friedman (ed.), Human values and the design of computer technology. Center for the Study of Language and Information, Stanford University, Stanford, Calif., in press.

14. Roth, A. E. The evolution of the labor market for medical interns and residents: A case study in game theory. Journal of Political Economy 92 (1984): 991–1016.

15. Roth, A. E. New physicians: A natural experiment in market organization. Science 250 (1990): 1524–1528.

16. Roth, S. K. The unconsidered ballot: How design effects voting behavior. Visible Language 28 (1994): 48–67.

17. Shifrin, C. A. Justice will weigh suit challenging airlines' computer reservations. Aviation Week & Space Technology (March 1985), p. 105.

18. Shneiderman, B. and Rose, A. Social impact statements: Engaging public participation in information technology design. In B. Friedman (ed.), Human values and the design of computer technology. Center for the Study of Language and Information, Stanford University, Stanford, Calif., in press.

19. Taib, I. M. Loophole allows bias in displays on computer reservations systems. Aviation Week & Space Technology (February 1990), p. 137.

20. Tang, J. C. Eliminating a hardware switch: Weighing economics and values in a design decision. In B. Friedman (ed.), Human values and the design of computer technology. Center for the Study of Language and Information, Stanford University, Stanford, Calif., in press.

21. Thomas, J. C. Steps toward universal access within a communications company. In B. Friedman (ed.), Human values and the design of computer technology. Center for the Study of Language and Information, Stanford University, Stanford, Calif., in press.