Bias in Computer Systems

BATYA FRIEDMAN
Colby College and The Mina Institute
and
HELEN NISSENBAUM
Princeton University

From an analysis of actual cases, three categories of bias in computer systems have been developed: preexisting, technical, and emergent. Preexisting bias has its roots in social institutions, practices, and attitudes. Technical bias arises from technical constraints or considerations. Emergent bias arises in a context of use. Although others have pointed to bias in particular computer systems and have noted the general problem, we know of no comparable work that examines this phenomenon comprehensively and which offers a framework for understanding and remedying it. We conclude by suggesting that freedom from bias should be counted among the select set of criteria—including reliability, accuracy, and efficiency—according to which the quality of systems in use in society should be judged.

Categories and Subject Descriptors: D.2.0 [**Software**]: Software Engineering; H.1.2 [**Information Systems**]: User/Machine Systems; K.4.0 [**Computers and Society**]: General

General Terms: Design, Human Factors

Additional Key Words and Phrases: Bias, computer ethics, computers and society, design methods, ethics, human values, standards, social computing, social impact, system design, universal design, values

INTRODUCTION

To introduce what bias in computer systems might look like, consider the case of computerized airline reservation systems, which are used widely by travel agents to identify and reserve airline flights for their customers. These reservation systems seem straightforward. When a travel agent types in a customer's travel requirements, the reservation system searches

This research was funded in part by the Clare Boothe Luce Foundation.
Earlier aspects of this work were presented at the 4S/EASST Conference, Goteborg, Sweden, August 1992, and at InterCHI '93, Amsterdam, April 1993. An earlier version of this article appeared as Tech. Rep. CSLI-94-188, CSLI, Stanford University.
Authors' addresses: B. Friedman, Department of Mathematics and Computer Science, Colby College, Waterville, ME 04901; email: b_friedm@colby.edu; H. Nissenbaum, University Center for Human Values, Marx Hall, Princeton University, Princeton, NJ 08544; email: helen@phoenix.princeton.edu.

a database of flights and retrieves all reasonable flight options that meet or come close to the customer's requirements. These options then are ranked according to various criteria, giving priority to nonstop flights, more direct routes, and minimal total travel time. The ranked flight options are displayed for the travel agent. In the 1980s, however, most of the airlines brought before the Antitrust Division of the United States Justice Department allegations of anticompetitive practices by American and United Airlines whose reservation systems—Sabre and Apollo, respectively—dominated the field. It was claimed, among other things, that the two reservations systems are biased [Schrifin 1985].

One source of this alleged bias lies in Sabre's and Apollo's algorithms for controlling search and display functions. In the algorithms, preference is given to "on-line" flights, that is, flights with all segments on a single carrier. Imagine, then, a traveler who originates in Phoenix and flies the first segment of a round-trip overseas journey to London on American Airlines, changing planes in New York. All other things being equal, the British Airlines' flight from New York to London would be ranked lower than the American Airlines' flight from New York to London even though in both cases a traveler is similarly inconvenienced by changing planes and checking through customs. Thus, the computer systems systematically downgrade and, hence, are biased against international carriers who fly few, if any, internal U.S. flights, and against internal carriers who do not fly international flights [Fotos 1988; Ott 1988].

Critics also have been concerned with two other problems. One is that the interface design compounds the bias in the reservation systems. Lists of ranked flight options are displayed screen by screen. Each screen displays only two to five options. The advantage to a carrier of having its flights shown on the first screen is enormous since 90% of the tickets booked by travel agents are booked by the first screen display [Taib 1990]. Even if the biased algorithm and interface give only a small percent advantage overall to one airline, it can make the difference to its competitors between survival and bankruptcy. A second problem arises from the travelers' perspective. When travelers contract with an independent third party—a travel agent—to determine travel plans, travelers have good reason to assume they are being informed accurately of their travel options; in many situations, that does not happen.

As Sabre and Apollo illustrate, biases in computer systems can be difficult to identify let alone remedy because of the way the technology engages and extenuates them. Computer systems, for instance, are comparatively inexpensive to disseminate, and thus, once developed, a biased system has the potential for widespread impact. If the system becomes a standard in the field, the bias becomes pervasive. If the system is complex, and most are, biases can remain hidden in the code, difficult to pinpoint or explicate, and not necessarily disclosed to users or their clients. Furthermore, unlike in our dealings with biased individuals with whom a potential victim can negotiate, biased systems offer no equivalent means for appeal.

Although others have pointed to bias in particular computer systems and have noted the general problem [Johnson and Mulvey 1993; Moor 1985], we know of no comparable work that focuses exclusively on this phenomenon and examines it comprehensively.

In this article, we provide a framework for understanding bias in computer systems. From an analysis of actual computer systems, we have developed three categories: preexisting bias, technical bias, and emergent bias. Preexisting bias has its roots in social institutions, practices, and attitudes. Technical bias arises from technical constraints or considerations. Emergent bias arises in a context of use. We begin by defining bias and explicating each category and then move to case studies. We conclude with remarks about how bias in computer systems can be remedied.

1. WHAT IS A BIASED COMPUTER SYSTEM?

In its most general sense, the term bias means simply "slant." Given this undifferentiated usage, at times the term is applied with relatively neutral content. A grocery shopper, for example, can be "biased" by not buying damaged fruit. At other times, the term bias is applied with significant moral meaning. An employer, for example, can be "biased" by refusing to hire minorities. In this article we focus on instances of the latter, for if one wants to develop criteria for judging the quality of systems in use—which we do—then criteria must be delineated in ways that speak robustly yet precisely to relevant social matters. Focusing on bias of moral import does just that.

Accordingly, we use the term bias to refer to computer systems that *systematically* and *unfairly discriminate* against certain individuals or groups of individuals in favor of others. A system 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. Consider, for example, an automated credit advisor that assists in the decision of whether or not to extend credit to a particular applicant. If the advisor denies credit to individuals with consistently poor payment records we do not judge the system to be biased because it is reasonable and appropriate for a credit company to want to avoid extending credit privileges to people who consistently do not pay their bills. In contrast, a credit advisor that systematically assigns poor credit ratings to individuals with ethnic surnames discriminates on grounds that are not relevant to credit assessments and, hence, discriminates unfairly.

Two points follow. First, unfair discrimination alone does not give rise to bias unless it occurs systematically. Consider again the automated credit advisor. Imagine a random glitch in the system which changes in an isolated case information in a copy of the credit record for an applicant who happens to have an ethnic surname. The change in information causes a downgrading of this applicant's rating. While this applicant experiences unfair discrimination resulting from this random glitch, the applicant could have been anybody. In a repeat incident, the same applicant or others with similar ethnicity would not be in a special position to be singled out. Thus, while the system is prone to random error, it is not biased.

Second, systematic discrimination does not establish bias unless it is joined with an unfair outcome. A case in point is the Persian Gulf War, where United States Patriot missiles were used to detect and intercept Iraqi Scud missiles. At least one software error identified during the war contributed to systematically poor performance by the Patriots [Gao 1992]. Calculations used to predict the location of a Scud depended in complex ways on the Patriots' internal clock. The longer the Patriot's continuous running time, the greater the imprecision in the calculation. The deaths of at least 28 Americans in Dhahran can be traced to this software error, which systematically degraded the accuracy of Patriot missiles. While we are not minimizing the serious consequence of this systematic computer error, it falls outside of our analysis because it does not involve unfairness.

2. FRAMEWORK FOR ANALYZING BIAS IN COMPUTER SYSTEMS

We derived our framework by examining actual computer systems for bias. Instances of bias were identified and characterized according to their source, and then the characterizations were generalized to more abstract categories. These categories were further refined by their application to other instances of bias in the same or additional computer systems. In most cases, our knowledge of particular systems came from the published literature. In total, we examined 17 computer systems from diverse fields including banking, commerce, computer science, education, medicine, and law.

The framework that emerged from this methodology is comprised of three overarching categories—preexisting bias, technical bias, and emergent bias. Table I contains a detailed description of each category. In more general terms, they can be described as follows.

2.1 Preexisting Bias

Preexisting bias has its roots in social institutions, practices, and attitudes. When computer systems embody biases that exist independently, and usually prior to the creation of the system, then we say that the system embodies preexisting bias. Preexisting biases may originate in society at large, in subcultures, and in formal or informal, private or public organizations and institutions. They can also reflect the personal biases of individuals who have significant input into the design of the system, such as the client or system designer. This type of bias can enter a system either through the explicit and conscious efforts of individuals or institutions, or implicitly and unconsciously, even in spite of the best of intentions. For example, imagine an expert system that advises on loan applications. In determining an applicant's credit risk, the automated loan advisor negatively weights applicants who live in "undesirable" locations, such as low-income or high-crime neighborhoods, as indicated by their home addresses (a practice referred to as "red-lining"). To the extent the program

Table I. Categories of Bias in Computer System Design

These categories describe ways in which bias can arise in the design of computer systems. The illustrative examples portray plausible cases of bias.

1. Preexisting Bias
Preexisting bias has its roots in social institutions, practices, and attitudes.
When computer systems embody biases that exist independently, and usually prior to the creation of the system, then the system exemplifies preexisting bias. Preexisting bias can enter a system either through the explicit and conscious efforts of individuals or institutions, or implicitly and unconsciously, even in spite of the best of intentions.

1.1. Individual
Bias that originates from individuals who have significant input into the design of the system, such as the client commissioning the design or the system designer (e.g., a client embeds personal racial biases into the specifications for loan approval software).

1.2 Societal
Bias that originates from society at large, such as from organizations (e.g., industry), institutions (e.g., legal systems), or culture at large (e.g., gender biases present in the larger society that lead to the development of educational software that overall appeals more to boys than girls).

2. Technical Bias
Technical bias arises from technical constraints or technical considerations.

2.1 Computer Tools
Bias that originates from a limitation of the computer technology including hardware, software, and peripherals (e.g., in a database for matching organ donors with potential transplant recipients certain individuals retrieved and displayed on initial screens are favored systematically for a match over individuals displayed on later screens).

2.2 Decontextualized Algorithms
Bias that originates from the use of an algorithm that fails to treat all groups fairly under all significant conditions (e.g., a scheduling algorithm that schedules airplanes for take-off relies on the alphabetic listing of the airlines to rank order flights ready within a given period of time).

2.3 Random Number Generation
Bias that originates from imperfections in pseudorandom number generation or in the misuse of pseudorandom numbers (e.g., an imperfection in a random-number generator used to select recipients for a scarce drug leads systematically to favoring individuals toward the end of the database).

2.4 Formalization of Human Constructs
Bias that originates from attempts to make human constructs such as discourse, judgments, or intuitions amenable to computers: when we quantify the qualitative, discretize the continuous, or formalize the nonformal (e.g., a legal expert system advises defendants on whether or not to plea bargain by assuming that law can be spelled out in an unambiguous manner that is not subject to human and humane interpretations in context).

Table I. *Continued*

These categories describe ways in which bias can arise in the design of computer systems. The illustrative examples portray plausible cases of bias.
3. Emergent Bias Emergent bias arises in a context of use with real users. This bias typically emerges some time after a design is completed, as a result of changing societal knowledge, population, or cultural values. User interfaces are likely to be particularly prone to emergent bias because interfaces by design seek to reflect the capacities, character, and habits of prospective users. Thus, a shift in context of use may well create difficulties for a new set of users.
3.1 New Societal Knowledge Bias that originates from the emergence of new knowledge in society that cannot be or is not incorporated into the system design (e.g., a medical expert system for AIDS patients has no mechanism for incorporating cutting-edge medical discoveries that affect how individuals with certain symptoms should be treated).
3.2 Mismatch between Users and System Design Bias that originates when the population using the system differs on some significant dimension from the population assumed as users in the design.
3.2.1 Different Expertise Bias that originates when the system is used by a population with a different knowledge base from that assumed in the design (e.g., an ATM with an interface that makes extensive use of written instructions—“place the card, magnetic tape side down, in the slot to your left”—is installed in a neighborhood with primarily a nonliterate population).
3.2.2 Different Values Bias that originates when the system is used by a population with different values than those assumed in the design (e.g., educational software to teach mathematics concepts is embedded in a game situation that rewards individualistic and competitive strategies, but is used by students with a cultural background that largely eschews competition and instead promotes cooperative endeavors).

embeds the biases of clients or designers who seek to avoid certain applicants on the basis of group stereotypes, the automated loan advisor's bias is preexisting.

2.2 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 pseudorandom number generation; and the attempt to make human constructs amenable to computers, when we quantify the qualitative, discretize the continuous, or formalize the nonformal. As an illustration, consider again the case of Sabre and Apollo described above. A technical constraint imposed by the size of the monitor screen forces a piecemeal presentation of flight options and, thus, makes the algorithm chosen to rank flight options critically important. Whatever ranking algorithm is used, if it systematically places certain airlines' flights on initial screens and other airlines' flights on later screens, the system will exhibit technical bias.

2.3 Emergent Bias

While it is almost always possible to identify preexisting bias and technical bias in a system design at the time of creation or implementation, emergent bias arises only in a context of use. This bias typically emerges some time after a design is completed, as a result of changing societal knowledge, population, or cultural values. Using the example of an automated airline reservation system, envision a hypothetical system designed for a group of airlines all of whom serve national routes. Consider what might occur if that system was extended to include international airlines. A flight-ranking algorithm that favors on-line flights when applied in the original context with national airlines leads to no systematic unfairness. However, in the new context with international airlines, the automated system would place these airlines at a disadvantage and, thus, comprise a case of emergent bias. User interfaces are likely to be particularly prone to emergent bias because interfaces by design seek to reflect the capacities, character, and habits of prospective users. Thus, a shift in context of use may well create difficulties for a new set of users.

3. APPLICATIONS OF THE FRAMEWORK

We now analyze actual computer systems in terms of the framework introduced above. It should be understood that the systems we analyze are by and large good ones, and our intention is not to undermine their integrity. Rather, our intention is to develop the framework, show how it can identify and clarify our understanding of bias in computer systems, and establish its robustness through real-world cases.

3.1 The National Resident Match Program (NRMP)

The NRMP implements a centralized method for assigning medical school graduates their first employment following graduation. The centralized method of assigning medical students to hospital programs arose in the 1950s in response to the chaotic job placement process and on-going failure of hospitals and students to arrive at optimal placements. During this early period the matching was carried out by a mechanical card-sorting process, but in 1974 electronic data processing was introduced to handle the entire matching process. (For a history of the NRMP, see Graettinger and Peranson [1981a].) After reviewing applications and interviewing students, hospital programs submit to the centralized program their ranked list of students. Students do the same for hospital programs. Hospitals and students are not permitted to make other arrangements with one another or to attempt to directly influence each others' rankings prior to the match.

Table II. The Simplest Case of Rank-Order Lists in which the Desires of Students and Programs are in Conflict (Reprinted from Williams et al. [1991], by permission of the *New England Journal of Medicine*)

Rank Order	Student I	Student II	Program A	Program B
1	Program A	Program B	Student II	Student I
2	Program B	Program A	Student I	Student II

Note: Each program in this example has a quota of one position.

With the inputs from hospitals and students, the NRMP applies its "Admissions Algorithm" to produce a match.

Over the years, the NRMP has been the subject of various criticisms. One charges that the Admissions Algorithm systematically favors hospital programs over medical students in cases of conflict [Graettinger and Peranson 1981b; Roth 1984; Sudarshan and Zisook 1981; Williams et al. 1981]. Consider the example developed by Williams et al., which we reproduce:

> To generate a match from Table II, the NRMP algorithm first attempts a so-called 1:1 run, in which concordances between the first choice of students and programs are matched (this table was constructed so that there would be none). The algorithm then moves to a 2:1 run, in which the students' second choices are tentatively run against the programs' first choices. Both students are matched with their second-choice programs. This tentative run becomes final, since no students or program is left unmatched. Matching is completed; both programs receive their first choices, and both students their second choices.
>
> The result of switching the positions of the students and programs in the algorithm should be obvious. After the 1:1 run fails, the 2:1 run under a switch would tentatively run the programs' second choices against the students' first choices, thus matching both programs with their second-choice students. Matching is again completed, but on this run, both students receive their first choices, and the programs . . . receive their second choices [Williams et al. 1981, p. 1165].

Does such preference for hospital programs reflect bias? We are inclined to answer yes because in cases of conflict there does not appear to be a good rationale for favoring hospital programs at the expense of students. Moreover, Graettinger and Peranson provide grounds for assessing the type of bias. They write, "The constraint inherent in the NRMP algorithm, in which preference is given to hospital choices when conflicts in rankings occur, duplicates what happens in an actual admissions process without a computerized matching program" [Graettinger and Peranson 1981b, p. 526]. Elsewhere, they write:

> Changing the [Admissions] algorithm would imply changing the NRMP's role from one of passive facilitator to one in which the NRMP would be intervening in the admissions process by imposing a different

> result than would be obtained without the matching program. This is not the role for which the NRMP was intended [p. 526].

Thus, if the algorithm systematically and unfairly favors hospital over student preferences, it does so because of design specifications and organizational practices that predate the computer implementation. As such, the NRMP embodies a preexisting bias.

Earlier versions of the NRMP have also been charged with bias against married couples in cases where both the husband and wife were medical students. When the NRMP was originally designed, few such couples participated in the medical match process. Beginning, however, in the late 1970s and early 1980s more women entered medical schools, and not surprisingly, more married couples sought medical appointments through the NRMP. At this point, it was discovered that the original Admissions Algorithm placed married couples at a disadvantage in achieving an optimal placement as compared with their single peers [Roth 1984; 1990]. Roth describes the problem as follows:

> Prior to the mid-1980s, couples participating in the match were required to specify one of their members as the "leading member," and to submit a rank ordering of positions for each member of the couple; that is, a couple submitted two preference lists. The leading member was matched to a position in the usual way, the preference list of the other member of the couple was edited to remove distant positions, and the second member was then matched if possible to a position in the same vicinity as the leading member. It is easy to see how instabilities [nonoptimum matches] would often result. Consider a couple whose first choice is to have two particular jobs in Boston, and whose second choice is to have two particular jobs in New York. Under the couples algorithm, the leading member might be matched to his or her first choice job in Boston, whereas the other member might be matched to some undesirable job in Boston. If their preferred New York jobs ranked this couple higher than students matched to those jobs, an instability would now exist [Roth 1990, p. 1528].

In this example, once the leading member of the couple is assigned a match in Boston no other geographic locations for the couple are considered. Thus, a better overall match with a hospital in New York is missed. The point here is that the bias—in this case emergent bias—against couples primarily emerged when a shift occurred in the social conditions, namely, when husband and wife medical students increasingly participated in the match process.

Compare the above two charges of bias with a third one, which accuses the NRMP of bias against hospitals in rural areas because of a consistent placement pattern over the years in which urban hospitals are far more successful in filling their positions than rural ones [Roth 1984; Sudarshan and Zisook 1981]. The Admissions Algorithm does not take into account geographic distribution when determining a match, considering only the ranked preferences of hospitals and students. Because the best teaching hospitals tend to be in urban areas, the urban areas tend to fill their positions far more effectively, and with better students, than rural hospitals. Observing this uneven distribution some have concluded that the NRMP is biased against rural hospitals in favor of urban ones. Is this so?

While we are committed to a stance against injustice, we do not think the distinction between a rationally based discrimination and bias is always easy to draw. In some cases, reasonable people might differ in their judgments. In this case, we ourselves would shy away from viewing this as a bias in the system because we think this discrimination can be defended as having a reasonable basis. Namely, the discrimination reflects the preferences of match participants, and it is reasonable in our view for employment decisions to be determined largely by the choices of employers and employees.

Bias in the NRMP is particularly troubling because of the system's centralized status. Most major hospitals agree to fill their positions with the NRMP assignments. Thus, for an individual or couple to elect not to participate in the NRMP is tantamount to forgoing the possibility of placement in most hospital programs. In this manner, centralized computing systems with widespread use can hold users hostage to whatever biases are embedded within the system.

3.2 A Multilevel Scheduling Algorithm (MLSA)

In timeshare computer systems many individuals make use of a single computer. These systems face a common problem of how to schedule the processing demands of the many individuals who make use of the processor at the same time. When posed with how to share equitably a limited resource, accepted social practice often points toward a first-come, first-serve basis. But the practical shortcomings of this approach are readily apparent. Imagine a person who uses the computer for interactive editing. Such work entails many small jobs that take very little time to process. Should another user with a large job requiring several minutes or hours of computation come along, the first user would experience a noticeable delay between execution of editing commands. Likely enough, frustrations would run high, and users would be dissatisfied. Thus, a balance must be struck between providing a reasonable response time and relatively efficient computation speed of large and long-running programs.

Proposed by F. J. Corbato in the 1960's, the MLSA represents one algorithm to address this balance [Corbato et al. 1962]. This algorithm was implemented in the CTSS timesharing system and in Multics. In brief, the MLSA works as follows. When a new command is received, it is executed for up to a quantum of time. If the process is not completed in that quantum of time, then the process is placed in a queue for "longer-running processes." Other new commands, if present, then are processed. Only when there are no new commands does the processor return to the process left in the queue for longer-running processes. Execution of this process is continued for a larger quantum of time. If the process is not completed in this larger quantum of time, then it is placed in yet another queue of "even longer running processes." And again, the processor returns to execute any new commands and, after that, any processes in the queue for longer-running processes. Only when there are no new commands and the queue for longer-running processes is empty will the processor look to the queue of even longer running processes for unfinished processes. In this manner the MLSA gives processing attention to all processes as quickly as possible that are beginning a new command. Thus, assuming the system is not saturated with too many users, short-running processes are speedily processed to completion. At the same time, however, in principle a long-running process could wait all day to finish.

Does the balance between response time and computation speed of long-running programs achieved by the MLSA systematically disadvantage some users? To help answer this question, consider a situation in which many people use a timeshare system at the same time on a regular basis. Of these individuals, most use relatively small programs on relatively short tasks, such as the interactive editing mentioned above. However, one or two individuals consistently use the system to execute long-running programs. According to the MLSA, the long-running programs of these individuals will necessarily end up with a lower priority than the short-running tasks of the other users. Thus, in terms of overall service from the processor, these individuals with long-running programs are systematically disadvantaged. According to Corbato (electronic communication, December 17, 1993), in response to this situation, some users with long-running programs uncovered the MLSA's strategy and developed a counterstrategy: by using a manual button to stop execution of a long-running process and a moment later restarting the process from where it left off, users effectively ran their long-running tasks in small chunks. Each small chunk, of course, was placed by MLSA into the top priority queue and executed speedily.

Having established systematic discrimination in the MLSA, we next ask whether this systematic discrimination is unfair. Consider that in other sorts of mundane queuing situations, such as movies or banks, we generally perceive that the "first-come first-served" strategy is fairest. It is also true that we can appreciate alternative strategies if they can complement and replace the strategy. In supermarkets, for example, we can appreciate express checkouts. But we likely would perceive as unfair a checkout system in which customers with fewer than, say, 10 items in their baskets could push ahead of anyone else in the line with more than ten items. Similarly, it seems to us that by systematically favoring short jobs, the MSLA violates the fairness preserved in the "first-come first-served" strategy.

While a human context gave rise to the need for the scheduling algorithm, it is important to understand that there was no prior bias against individuals with longer-running jobs. That is, the algorithm's bias did not arise from social factors, say, to dissuade users from large computational projects or to encourage interactive editing and debugging. Had this been the case, the bias would be preexisting. Rather the bias is technical, for the algorithm arose in the attempt to satisfy a difficult technical requirement to allocate a scare resource. It does so by giving processing attention to all processes as quickly as possible.

Another algorithm might have eluded the MLSA's form of technical bias by balancing response time and long-running computations in a manner that did not lead to systematic disadvantage for individuals with long-running programs. However, we also recognize that in the attempt to strike a balance between two apparently conflicting claims on a processor, it may not be possible to achieve a solution that is completely fair to all of those using the system. In cases like the MLSA, an awareness may be needed that one group is disadvantaged by the system, and an attempt made to minimize that disadvantage from within the system or to address it by some other means.

3.3 The British Nationality Act Program (BNAP)

Before discussing bias in the BNAP, a bit of history. In 1981 the Thatcher government passed the British Nationality Act as a means to redefine British citizenship. The act defined three new categories: British citizenship, citizenship of the British Dependent Territories, and British overseas citizenship. Only full British citizens in the first category would have the right to live in Britain. While the Thatcher government and some British citizens defended the act, others raised objections. For example, within Britain, according to *The London Times*, "The Labour Shadow Cabinet . . . decided to oppose the Bill on the grounds that it contains elements of racial and sexual discrimination" [Berlins and Hodges 1981, p. 1]. Similarly, in India, the *Hindustan Times* reported (quoted by Fishlock [1981]):

> Racial discrimination, by whatever name or device, is still discrimination of the most reprehensible kind. The Bill formalizes and legitimates racism toward people of a different hue which reflects the xenophobic paranoia that afflicts a section of British society today. The proposed three tiers of citizenship are a fine sieve which will allow into Britain only those of the desired racial stock.

Beginning in 1983, M. J. Sergot and his colleagues at the Imperial College, University of London, undertook to translate the British Nationality Act into a computer program so that "consequences of the act can be determined mechanistically" [Sergot et al. 1986, p. 370].[1] Critics have charged BNAP of gender bias. Consider the following. One of the most compelling grounds for establishing British citizenship is to have at least one parent who is a British citizen. As specified in Section 50-(9) of the British Nationality Act itself and imple-

[1] Although the program authors [Sergot et al. 1986] state "It was never our intention to develop the implementation of the act into a fully functional program" (p. 371), it is difficult to take their disclaimer entirely seriously. For in the same text the authors also state (as noted above) that their goal is to translate the British Nationality Act so that "the consequences of the act can be determined mechanistically" (p. 370), and they place their work in the context of other legal expert systems designed for use with real users.

mented in BNAP, “a man is the ‘father’ of only his legitimate children, whereas a woman is the ‘mother’ of all her children, legitimate or not” [Sergot et al. 1986, p. 375]. Consider then the instance of an unmarried man and woman who live together with the children they have jointly conceived. If the mother is a British citizen, and the father is not, then the children have one parent who is considered a British citizen. But if the situation is reversed (that is, if the father is a British citizen, and the mother is not), then the children have no parents who are considered British citizens. Thus the British Nationality Act is biased against the illegitimate descendants of British men. Accordingly, to the extent the BNAP accurately represents the British Nationality Act, the program embodies preexisting bias.

Two further concerns with the BNAP can be understood in terms of emergent bias. First, the system was designed in a research environment, among people with sophisticated knowledge of immigration law. Its users, however, are likely to be at best paralegal or immigration counselors in Britain, if not lay persons in foreign countries with only limited access to British legal expertise. A problem thus arises for nonexpert users. Some of the program’s queries, for example, require expert knowledge to answer. More generally, nonexperts advising British citizen hopefuls are not alerted to alternative legal frameworks that complement the British Nationality Act. Thus nonexperts—particularly in developing countries—would be inclined to accept decisively the BNAP’s response to their client’s situation. In such ways, the BNAP comes to act as an instrument of bias against the nonexperts and their clients. Because this bias arises from a shift in the population using the system from the one apparently held in mind by the system’s creators (from expert to nonexpert users) we identify this bias as emergent.

Another source for emergent bias can arise in the following way. At the time of the BNAP’s initial implementation (1983) no mechanism was built into the program to incorporate relevant case law as it came into being [Sergot et al. 1986]. Should the accumulation of case law lead to changes in the way the Act is interpreted—say, by granting new subgroups British citizenship—BNAP would systematically misinform members of this subgroup regarding their status as British citizens [Leith 1986]. Again, we identify this bias as emergent because it depends on a shift in the social context, in this instance one concerning knowledge within the legal community. Emergent bias poses a potential problem for any legal expert system, especially in a society where legal systems depend on evolving case law. Indeed, to varying degrees, any expert system (independent of content area) that does not possess a reasonable mechanism for integrating new knowledge may be vulnerable to a similar form of emergent bias.

4. CONSIDERATIONS FOR MINIMIZING BIAS IN COMPUTER SYSTEM DESIGN

As our framework helps delineate the problems of bias in computer systems, so does it offer ways to remedy them. But, before saying more along these lines, it is important to address two potential concerns that the reader may have about the framework itself.

First, as we have noted earlier, computer systems sometimes help implement social policies on which reasonable people can disagree regarding whether the policies are fair or unfair. Does the NRMP, for example, embody a bias against rural hospitals? Other examples might include affirmative-action hiring programs, tax laws, and some federal funding programs. According to our framework, would computer systems that help implement such discriminative policies embody bias? The answer follows the initial (controversial) question—namely, "Is the policy under consideration fair or unfair?" Does affirmative action, for example, help redress past unfairness or not? The answer to most of these questions is beyond the scope of this article. But we do say that if unfairness can be established in the system's systematic discrimination, then the charge of bias follows.

Second, although we have talked about bias in computer systems, the presence of bias is not so much a feature inherent in the system independent of the context of use, but an aspect of a system in use. This distinction can be seen clearly in an example of emergent bias. Consider the case of an intelligent tutoring system on AIDS whose intended users are to be college students. Here, a high degree of literacy can be assumed without incurring bias. In contrast, the same level of literacy cannot be assumed without introducing bias in designing a system to provide AIDS education in a public space such as a shopping mall or metro station. For in such public spaces less educated people would be at an unfair disadvantage in using the system. Or consider again the case of technical bias with the MLSA (which favors users with short jobs over long jobs). While technical bias is embedded in the program, the bias is a phenomenon of the system in use, in a context wherein users with short and long jobs outpace the system's capacity.

Remedying bias from a practical perspective involves at least two types of activities. One, we need to be able to identify or "diagnose" bias in any given system. Second, we need to develop methods of avoiding bias in systems and correcting it when it is identified. We offer below some initial directions this work could take.

Toward minimizing preexisting bias, designers must not only scrutinize the design specifications, but must couple this scrutiny with a good understanding of relevant biases out in the world. The time to begin thinking about bias is in the earliest stages of the design process, when negotiating the system's specifications with the client. Common biases might occur to populations based on cultural identity, class, gender, literacy (literate/less literate), handedness (right-handed/left-handed), and physical disabilities (e.g., being blind, color-blind, or deaf). As the computing community develops an understanding of these biases, we can correspondingly develop techniques to avoid or minimize them. Some current computer systems, for instance, address the problem of handedness by allowing the user to toggle between a right- or left-handed configuration for user input and screen display. Similarly, systems could minimize bias due to color blindness by encoding information not only in hue, but in its intensity, or in some other way by encoding the same information in a format unrelated to color. In addition, it can prove useful to identify potential user populations which might otherwise be overlooked and include representative individuals in the field test groups. Rapid prototyping, formative evaluation, and field testing with such well-conceived populations of users can be an effective means to detect unintentional biases throughout the design process.

Technical bias also places the demand on a designer to look beyond the features internal to a system and envision it in a context of use. Toward preventing technical bias, a designer must envision the design, the algorithms, and the interfaces in use so that technical decisions do not run at odds with moral values. Consider even the largely straightforward problem of whether to display a list with random entries or sorted alphabetically. In determining a solution, a designer might need to weigh considerations of ease of access enhanced by a sorted list against equity of access supported by a random list.

Minimizing emergent bias asks designers to envision not only a system's intended situations of use, but to account for increasingly diverse social contexts of use. From a practical standpoint, however, such a proposal cannot be pursued in an unbounded manner. Thus, how much diversity in social context is enough, and what sort of diversity? While the question merits a lengthy discussion, we offer here but three suggestions. First, designers should reasonably anticipate probable contexts of use and design for these. Second, where it is not possible to design for extended contexts of use, designers should attempt to articulate constraints on the appropriate contexts of a system's use. As with other media, we may need to develop conventions for communicating the perspectives and audience assumed in the design. Thus, if a particular expert system because of its content matter, goals, and design requires expert users to be used effectively, this constraint should be stated clearly in a salient place and manner, say, on one of the initial screens. Third, system designers and administrators can take responsible action if bias emerges with changes in context. The NRMP offers a good example. Although the original design of the Admissions Algorithm did not deal well with the changing social conditions (when significant numbers of dual-career couples participated in the match), those responsible for maintaining the system responded conscientiously to this societal change and modified the system's algorithm to place couples more fairly [Roth 1990].

That said, even if a designer successfully detects bias in a proposed design, and has ideas on how to eradicate or minimize it, a client may be reluctant to remedy the bias for a variety of reasons. For example, airlines executives whose companies serve national and international routes may knowingly support the bias in an automated reservation system that favors on-line flights. Situations such as these put designers in a difficult spot. What ought they to do if a client actually wants a bias to be present? Is it the designer's responsibility to speak out against a client, or is it simply to tow the line and produce a system, bias and all?

Readers who have followed discussions of professional ethics will be familiar with similar dilemmas. A quick answer is not possible, but one thing is clear. In order for designers to take an effective stand against a client regarding biased systems, it will be important for designers to find support for such action from their professional community. The criteria of reliability and safety offer a perspective on this point. Through extensive discussion and solid technical work, the computing community over the recent years has recognized that good systems must be judged on criteria that include reliability and safety. Such consensus provides individual system designers with substantial backing if and when they are required to make their cases to skeptical clients or employers. Something similar is needed for bias. The more the computing community explicitly recognizes bias as a feature of computer systems that is worth addressing and minimizing, the more individual designers will have clout in shaping the practice of computing in the work place and elsewhere.

While we advocate serious attention to bias in system design, we also recognize there are limits to what system designers can accomplish. Some biases extend beyond computing to larger societal problems. An empirical result from work by Huff and Cooper [1987] on gender-bias in the design of educational software provides a helpful illustration. In their 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 report 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. This study illustrates how preexisting biases, in the form of expectations about what 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. Huff and Cooper report, furthermore, that many of their subjects were aware of and had expressed open concern about how computers were frequently perceived to be in the male domain. We thus infer that in this case the biased designs were unintended and, instead, reflected gender-biases deeply rooted in the culture at large. While creating nongender-biased educational software contributes to addressing the larger social problems tied to gender bias and computing, resolving problems like gender bias go beyond system design. More broadly, where biases in the larger society flourish, bias-free system design forms but one part of a movement to a more equitable society.

5. CONCLUSION

Because biased computer systems are instruments of injustice—though admittedly, their degree of seriousness can vary considerably—we believe that freedom from bias should be counted among the select set of criteria according to which the quality of systems in use in society should be judged. The methods delineated here can be used to assess and minimize bias in the design of systems. Concern with bias in system design and experience with these methods can be integrated with other software engineering methods as part of the standard for a computer science curriculum.

As with other criteria for good computer systems, such as reliability, accuracy, and efficiency, freedom from bias should be held out as an ideal. As with these other criteria, this ideal might be difficult if not impossible to achieve. Nonetheless, in practice we must approach actively the task of minimizing bias in our designs. Furthermore, as a community we must hold our designs accountable to a reasonable degree of freedom from bias against which negligence can be judged.

ACKNOWLEDGMENTS

We thank our research assistant Mark Muir for help with aspects of this analysis. We thank F. J. Corbató, John Ellis, Deborah Johnson, James Moor, John Mulvey, Peter Neumann, and Alvin Roth for discussions concerning bias; and we thank Rob Kling and anonymous reviewers for their comments on this article. We extend our appreciation to CSLI, Stanford University, for use of their facilities during December, 1993.

REFERENCES

BERLINS, M. AND HODGES, L. 1981. Nationality Bill sets out three new citizenship categories. *The London Times* (Jan. 15), 1, 15.

CORBATÓ, F. J., MERWIN-DAGGETT, M., AND DALEY, R. C. 1962. An experimental time-sharing system. In *Proceedings of the Spring Joint Computer Conference.* Spartan Books, 335–344.

FISHLOCK, T. 1981. Delhi press detect racism in Nationality Bill. *The London Times* (Jan. 20).

FOTOS, C. P. 1988. British Airways assails U.S. decision to void CRS agreement with American. *Aviat. Week Space Tech.* (Oct. 24), 78.

GAO. 1992. Patriot Missile defense: Software problem led to system failure at Dhahran, Saudi Arabia. GAO/IMTEC-92-26, U.S. General Accounting Office, Washington, D.C.

GRAETTINGER, J. S. AND PERANSON, E. 1981a. The matching program. *New Engl. J. Med. 304,* 1163–1165.

GRAETTINGER, J. S. AND PERANSON, E. 1981b. National resident matching program. *New Engl. J. Med. 305,* 526.

HUFF, C. AND COOPER, J. 1987. Sex bias in educational software: The effect of designers' stereotypes on the software they design. *J. Appl. Soc. Psychol. 17,* 519–532.

JOHNSON, D. G. AND MULVEY, J. M. 1993. Computer decisions: Ethical issues of responsibility and bias. Statistics and Operations Res. Series SOR-93-11, Dept. of Civil Engineering and Operations Research, Princeton Univ., Princeton, N.J.

LEITH, P. 1986. Fundamental errors in legal logic programming. *Comput. J. 29,* 225–232.

MOOR, J. 1985. What is computer ethics? *Metaphilosophy 16,* 266–275.

OTT, J. 1988. American Airlines settles CRS dispute with British Airways. *Aviat. Week Space Tech.* (July 18).

ROTH, A. E. 1984. The evolution of the labor market for medical interns and residents: A case study in game theory. *J. Pol. Econ. 92,* 6, 991–1016.

ROTH, A. E. 1990. New physicians: A natural experiment in market organization. *Science 250,* (Dec. 14), 1524–1528.

SERGOT, M. J., SADRI, F., KOWALSKI, R. A., KRIWACZEK, F., HAMMOND, P., AND CORY, H. T. 1986. The British Nationality Act as a logic program. *Commun. ACM 29,* 370–386.

SHIFRIN, C. A. 1985. Justice will weigh suit challenging airlines' computer reservations. *Aviat. Week Space Tech.* (Mar. 25), 105–111.

SUDARSHAN, A. AND ZISOOK, S. 1981. National resident matching program. *New Engl. J. Med. 305,* 525–526.

TAIB, I. M. 1990. Loophole allows bias in displays on computer reservations systems. *Aviat. Week Space Tech.* (Feb.), 137.

WILLIAMS, K. J., WERTH, V. P., AND WOLFF, J. A. 1981. An analysis of the resident match. *New Engl. J. Med. 304,* 19, 1165–1166.

Received May 1994; revised February 1995; accepted August 1995