Science and Engineering Ethics (1996) **2**, 25-42

Accountability in a Computerized Society[1]

Helen Nissenbaum
University Center for Human Values, Princeton University, USA

Key Words: accountability, bugs, computer ethics, liability, moral responsibility, standard of care.

ABSTRACT: *This essay warns of eroding accountability in computerized societies. It argues that assumptions about computing and features of situations in which computers are produced create barriers to accountability. Drawing on philosophical analyses of moral blame and responsibility, four barriers are identified: 1) the problem of many hands, 2) the problem of bugs, 3) blaming the computer, and 4) software ownership without liability. The paper concludes with ideas on how to reverse this trend.*

> "If a builder has built a house for a man and has not made his work sound, and the house which he has built has fallen down and so caused the death of the householder, that builder shall be put to death.
> If it destroys property, he shall replace anything that it has destroyed; and, because he has not made sound the house which he has built and it has fallen down, he shall rebuild the house which has fallen down from his own property.
> If a builder has built a house for a man and does not make his work perfect and a wall bulges, that builder shall put that wall into sound condition at his own cost."
>
> Laws of Hammu-rabi [229, 232, 233][2], circa 2027 B.C.

Computing is an ongoing source of change in the way we conduct our lives. For the most part we judge these changes to be beneficial, but we also recognize that imperfections in the technology can, in significant measure, expose us to unexpected outcomes, as well as to harms and risks. Because the use of computing technology is so widespread these impacts are worrisome not only because harms can be severe, but because they pervade and threaten almost every sphere of public and private life. Lives and well-being are increasingly dependent on computerized life-critical systems that control aircraft (fly-by-wire), spacecraft, motor cars, military equipment,

Address for correspondence: Helen Nissenbaum, PhD, University Center for Human Values, Princeton University, Louis Marx Hall, Princeton NJ 08544-1606, USA.
Paper received 22 February 1995, revised 30 October 1995, accepted 17 November 1995.
1353-3452/96 © 1996 Opragen Publications, POB 54, Guildford GU1 2YF, England

communications devices and more. Quality of life is also at stake in the enormous array of information systems, communications networks, bureaucratic infrastructures of governments, corporations, and high finance, as well as everyday conveniences such as personal computers, telephones, microwaves and toys that are controlled and supported by computers.

The extensive presence of computing in these many spheres of life suggests two related concerns. The one is a concern with achieving a suitable degree of reliability and safety for these systems so as to minimize risks and harms; the other is a concern with entrenching and maintaining in those sectors of society that produce and purvey computing technologies a robust culture of accountability, or answerability, for their impacts. The first of these two has, in recent years, achieved increasing recognition among prominent members of the computer community[3]. They question whether many of the systems in use are sufficiently sound for the uses to which they are put. Citing cases of failure and poor programming practices, they appeal to the computer community[4], corporate producers, and government regulators, to pay more heed to system safety and reliability in order to reduce harms and risks [1-6] arguing that lives, well-being, and quality-of-life, are vulnerable to poor system design and the all too likely occurrence of failure.

But it is upon the second of these concerns, the concern for accountability, that this paper will focus. In the same way that experts within the computer community have exposed the critical need to improve standards of reliability for computer systems, this paper urges attention to the neglected status of accountability for the impacts of computing, specifically for the harms and risks of faulty and malfunctioning systems. Thus, while our vulnerability to system failure and risk argues for greater attention to system safety, reliability, and sound design, and calls for the development of technical strategies to achieve them, it also underscores the need for a robust tradition of *accountability* for failures, risks, and harm that do occur. A culture of accountability is particularly important for a technology still struggling with standards of reliability because it means that even in cases where things go awry, we are assured of answerability. However, just the opposite is occurring. This paper argues that conditions under which computer systems are commonly developed and deployed, coupled with popular conceptions about the nature, capacities, and limitations of computing, contribute in significant measure to an *obscuring* of lines of accountability. Unless we address these conditions and conceptions, we will see a disturbing correlation — increased computerization, on the one hand, with a decline in accountability, on the other.

A strong culture of accountability is worth pursuing for a number of reasons. For some, a developed sense of responsibility is a good in its own right, a virtue to be encouraged. Our social policies should reflect this value appropriately by expecting people to be accountable for their actions. For others, accountability is valued because of its consequences for social welfare. Firstly, holding people accountable for the harms or risks they bring about provides strong motivation for trying to prevent or minimize them. Accountability can therefore be a powerful tool for motivating better practices, and consequently more reliable and trustworthy systems. A general culture of accountability should encourage answerability not only for the life-critical systems

that cause or risk grave injuries, damage infrastructure, and cause large monetary losses, but even for the malfunctions that cause individual losses of time, convenience, and contentment. Secondly, maintaining clear lines of accountability means that in the event of harm through failure, we have a reasonable starting point for assigning just punishment as well as, where necessary, compensation for victims.

For the remainder of the paper I explain more fully the conditions in which computer systems are commonly produced and describe common assumptions about the capabilities and limitations of computing, showing how both contribute toward an erosion and obscuring of accountability. Four of these, which I henceforth call "the four barriers to accountability," will be the focus of most of the discussion. In identifying the barriers I hope at the same time to convince readers that as long as we fail to recognize and do something about these barriers to accountability, assigning responsibility for the impacts of computing will continue to be problematic in the many spheres of life that fall under its control. And unless we pursue means for reversing this erosion of accountability, there will be significant numbers of harms and risks for which no one is answerable and about which nothing is done. This will mean that computers may be "out of control"[5] in an important and disturbing way. I conclude the paper with brief remarks on how we might overcome the barriers and restore accountability.

Accountability, Blame, Responsibility — Conceptual Framework

The central thesis of this paper, that increasing computerization may come at the cost of accountability, rests on an intuitive understanding of accountability closely akin to "answerability." The following story captures its core in a setting that, I predict, will have a ring of familiarity to most readers.

> Imagine a teacher standing before her 6th-grade class demanding to know who shot a spit-ball in her ear. She threatens punishment for the whole class if someone does not step forward. Fidgety students avoid her stern gaze, as a boy in the back row slowly raises his hand.

This raising of his hand wherein the boy answers for his action signifies accountability. From the story alone, we do not know whether he shot at the teacher intentionally or merely missed his true target, whether he acted alone or under goading from classmates, or even whether the spit-ball was in protest for an unreasonable action taken by the teacher. While these factors may be relevant to determining a just response to the boy's action, we can say that the boy, in responding to the teacher's demand for an answer to who shot the spit-ball, has taken an important first step toward fulfilling the valuable social obligation of accountability. In this story, the boy in the back row has answered for, been accountable for, his action; in real life there can be conditions that obscure accountability.

For a deeper understanding of the barriers to accountability in a computerized society and the conditions that foster them, it is necessary to move beyond an intuitive grasp and to draw on ideas from philosophical and legal inquiry into moral responsibility and the cluster of interrelated concepts of liability, blame and accountability. Over the many years that these concepts have been discussed and

analyzed, both by those whose interest is theoretical in nature and those whose interest is more practical (Hammu-rabi's four thousand year old legal code is an early example), many analyses have been put forth, and many shadings of meaning have been discovered and described.

Emerging from this tradition, contemporary work by Joel Feinberg on moral blame provides a framework for this paper's inquiry[7]. Feinberg proposes a set of conditions under which an individual is morally blameworthy for a given harm[6]. Fault and causation are key conditions. Accordingly, a person is morally blameworthy for a harm if: 1) his or her actions caused the harm, or constituted a significant causal factor in bringing about the harm; and 2) his or her actions were "faulty"[7]. Feinberg develops the idea of faulty actions to cover actions that are guided by faulty decisions or intentions. This includes actions performed with an intention to hurt someone and actions for which someone fails to reckon adequately with harmful consequences. Included in the second group are reckless and negligent actions. We judge an action reckless if a person engages in it even though he foresees harm as its likely consequence but does nothing to prevent it; we judge it negligent, if he carelessly does not consider probable harmful consequences.

Applying Feinberg's framework to some examples, consider the case of a person who has intentionally installed a virus on someone's computer which causes extensive damage to files. This person is blameworthy because her intentional actions were causally responsible for the damage. In another case, one that actually occurred, Robert Morris, then a graduate student in computer science at Cornell University, whose Internet Worm caused major upheaval on the internet and infiltrated thousands of connected computers, was held blameworthy, even though the extensive damage was the consequence of a bug in his code and not directly intended. Critics judged him reckless because they contended that someone with Morris's degree of expertise ought to have foreseen this possibility[8].

Although moral blame is not identical to accountability, an important correspondence between the two makes the analysis of the former relevant to the study of the latter. An important set of cases in which one may reasonably expect accountability for a harm is that in which an analysis points to an individual (or group of individuals) who are morally blameworthy for it[9]. In these cases at least, moral blameworthiness provides a reasonable standard for answerability and, accordingly, Feinberg's conditions can be used to identify cases in which one would reasonably expect, or judge, that there ought to be accountability. The four barriers, explained in the sections below, are systematic features of situations in which we would reasonably expect accountability but for which accountability is obscured. For many situations of these types (though not all) the simplified version of Feinberg's analysis has helped bring into focus the source of breakdown.

The Problem of Many Hands[10]

Most computer systems in use today are the products not of single programmers working in isolation but of groups or organizations, typically corporations. These groups, which frequently bring together teams of individuals with a diverse range of

skills and varying degrees of expertise, might include designers, engineers, programmers, writers, psychologists, graphic artists, managers, and salespeople. Consequently, when a system malfunctions and gives rise to harm, the task of assigning responsibility—the problem of identifying who is accountable—is exacerbated and obscured. Responsibility, characteristically understood and traditionally analyzed in terms of a single individual, does not easily generalize to collective action. In other words, while the simplest quest for accountability would direct us in search of "the one" who must step forward (for example, the boy in the back row answering for the spit-ball), collective action presents a challenge. The analysis of blame, in terms of cause and fault, can help to clarify how in cases of collective action accountability can be lost, or at least, obscured.

Where a mishap is the work of "many hands," it may not be obvious who is to blame because frequently its most salient and immediate causal antecedents do not converge with its locus of decision making. The conditions for blame, therefore, are not clearly satisfied in a way normally satisfied when a single individual is held blameworthy for a harm. Indeed, some cynics argue that institutional structures are designed in this way precisely to avoid accountability. Furthermore, with the collective actions characteristic of corporate and government hierarchies, decisions and causes themselves are fractured. Team action, the endeavor of many individuals working together, creates a product which in turn causally interacts with the life and well-being of an end-user. Boards of directors, task forces, or committees issue joint decisions, and on the occasions where these decisions are not universally approved by all their members but are the result of majority vote, we are left with the further puzzle of how to attribute responsibility. When high-level decisions work their way down from boards of directors to managers, from managers to employees, ultimately translating into actions and consequences, the lines that bind a problem to its source may be convoluted and faint. And as a consequence the connection between an outcome and the one who is accountable for it is obscured. This obscuring of accountability can come about in different ways. In some cases, it may be the result of intentional planning, a conscious means applied by the leaders of an organization to avoid responsibility for negative outcomes, or it may be an unintended consequence of a hierarchical management in which individuals with the greatest decision-making powers are only distantly related to the causal outcome of their decisions. Whatever the reason, the upshot is that victims and those who represent them, are left without knowing at whom to point a finger. It may not be clear even to the members of the collective itself who is accountable. The problem of many hands is not unique to computing but plagues other technologies, big business, government, and the military [9-13].

Computing is particularly vulnerable to the obstacles of many hands. First, as noted earlier, most software systems in use are produced in institutional settings, including small and middle-sized software development companies, large corporations, government agencies and contractors, and educational institutions. Second, computer systems themselves, usually not monolithic, are constructed out of segments or modules. Each module itself may be the work of a team of individuals. Some systems may also include code from earlier versions, while others borrow code from different

systems entirely, even some that were created by other producers. When systems grow in this way, sometimes reaching huge and complex proportions, there may be no single individual who grasps the whole system or keeps track of all the individuals who have contributed to its various components [14,15]. Third, many systems being developed and already in use operate on top of other systems (such as intermediate level and special function programs and operating systems). Not only may these systems be unreliable, but there may merely be unforeseen incompatibilities between them[11]. Fourth, performance in a wide array of mundane and specialized computer-controlled machines—from rocket ships to refrigerators—depends on the symbiotic relationship of machine with computer system. When things go wrong, as shown below, it may be unclear whether the fault lies with the machine or with the computer system.

The case of the Therac-25, a computer-controlled radiation treatment machine, that massively overdosed patients in six known incidents[12] provides a striking example of the way many hands can obscure accountability. In the two-year period from 1985 to 1987, overdoses administered by the Therac-25 caused severe radiation burns which, in turn, caused death in three cases and irreversible injuries (one minor, two very serious) in the other three. Built by Atomic Energy of Canada Limited (AECL), Therac-25 was the further development in a line of medical linear accelerators which destroy cancerous tumors by irradiating them with accelerated electrons and X-ray photons. Computer controls were far more prominent in the Therac-25 both because the machine had been designed from the ground up with computer-controls in mind and also because the safety of the system as a whole was largely left to software. Whereas earlier models included hardware safety mechanisms and interlocks, designers of the Therac-25 did not duplicate software safety mechanisms with hardware equivalents.

After many months of study and trial-and-error testing, the origin of the malfunction was traced not to a single source, but to numerous faults, which included at least two significant software coding errors ("bugs") and a faulty microswitch[13]. The impact of these faults was exacerbated by the absence of hardware interlocks, obscure error messages, inadequate testing and quality assurance, exaggerated claims about the reliability of the system in AECL's safety analysis and, in at least two cases, negligence on the parts of the hospitals where treatment was administered. Aside from the important lessons in safety engineering that the Therac-25 case provides, it offers a lesson in accountability—or rather, the breakdown of accountability due to "many hands."

In cases like Therac-25, instead of identifying a single individual whose faulty actions caused the injuries, we find we must systematically unravel a messy web of interrelated causes and decisions. Even when we may safely rule out intentional wrongdoing it is not easy to pinpoint causal agents who were, at the same time, negligent or reckless. As a result, we might be forced to conclude that the mishaps were merely accidental[14] in the sense that no one can reasonably be held responsible, or to blame, for them. While a full understanding of the Therac-25 case would demand a more thorough study of the details than I can manage here, the sketch that follows is intended to show that though the conditions of many hands might indeed obscure accountability, they do not imply that answerability can be foregone.

Consider the many whose actions constituted causal antecedents of the Therac-25 injuries and in some cases contributed significantly to the existence and character of the machine. From AECL, we have designers, software and safety engineers, programmers, machinists, and corporate executives; from the clinics, we have administrators, physicians, physicists, and machine technicians. Take for example, those most proximately connected to the harm, the machine technicians who activated the Therac-25 by entering doses and pushing buttons. In one of the most chilling anecdotes associated with the Therac-25 incident, a machine technician is supposed to have responded to the agonized cries from a patient by flatly denying that it was possible that he had been burned. Should the blame be laid at her feet?

Except for specific incidents like the one involving the technician who denied a patient's screams of agony, accountability for the Therac-25 does not rest with the machine technicians because by and large they were not at fault in any way relevant to the harms and because the control they exercised over the machine's function was restricted to a highly limited spectrum of possibilities. By contrast, according to Leveson and Turner's discussion, there is clear evidence of inadequate software engineering, testing and risk assessment. For example, the safety analysis was faulty in that it systematically overestimated the system's reliability and evidently did not consider the role software failure could play in derailing the system as a whole. Moreover, computer code from earlier Therac models, used in the Therac-25 system, was assumed unproblematic because no similar malfunction had surfaced in these models. However, further investigation showed that while the problem *had* been present in those systems, it had simply not surfaced because earlier models had included mechanical interlocks which would override software commands leading to fatal levels of radiation. The Therac-25 did not include these mechanical interlocks.

There is also evidence of a failure in the extent of corporate response to the signs of a serious problem. Early response to reports of problems were particularly lackluster. AECL was slow to react to requests to check the machine, understand the problem, or to remediate (for example by installing an independent hardware safety system). Even after a patient filed a lawsuit in 1985 citing hospital, manufacturer, and service organization as responsible for her injuries, AECL's follow up was negligible. For example, no special effort was made to inform other clinics operating Therac-25 machines about the mishaps. Because the lawsuit was settled out of court, we do not learn how the law would have attributed liability.

Even Leveson and Turner, whose detailed analysis of the Therac-25 mishaps sheds light on both the technical as well as the procedural aspects of the case, hold back on the question of accountability. They refer to the malfunctions and injuries as "accidents" and remark that they do not wish "to criticize the manufacturer of the equipment or anyone else."[3] I mention this not as a strong critique of their work, because after all their central concern is unraveling the technical and design flaws in the Therac-25, but to raise the following point. Although a complex network of causes and decisions, typical of situations in which many hands operate, may obscure accountability, we ought not conclude that the harms were mere accidents. I have suggested that a number of individuals ought to have been answerable (though not in equal measure), from the machine operator who denied the possibility of burning to

the software engineers to quality assurance personnel and to corporate executives. Determining their degree of responsibility would require that we investigate more fully their *degree* of causal responsibility, control, and fault. By preferring to view the incidents as accidents, we may effectively be accepting them as agentless mishaps, yielding to the smoke-screen of collective action and to a further erosion of accountability.

The general lesson to be drawn from the case of the Therac-25 is that many hands obscured accountability by diminishing in key individuals a sense of responsibility for the mishaps. By contrast, a suitably placed individual (or several) ought to have stepped forward and assumed responsibility for the malfunction and harms. Instead, for two years, the problem bounced back and forth between clinics, manufacturer and various government oversight agencies, before concrete and decisive steps were taken. In collective action of this type, the plurality of causal antecedents and decision-makers helps to define a typical set of excuses for those low down in the hierarchy who are "only following orders," and for those of higher rank who are more distantly related to the outcomes. However, we should not mistakenly conclude from the observation that accountability is *obscured* due to collective action that no one is, or ought to have been, accountable. The worry that this paper addresses is that if computer technology is increasingly produced by "many hands," and if, as seems to be endemic to many hands situations, we lose touch with who is accountable (such as occurred with the Therac-25), then we are apt to discover a disconcerting array of computers in use for which no one is answerable.

Bugs

The source of a second barrier to accountability in computing is omnipresent bugs and the way many in the field routinely have come to view them. To say that bugs in software make software unreliable and cause systems to fail is to state the obvious. However, not quite as obvious is how the way we *think* about bugs affects considerations of accountability. (I use the term "bug" to cover a variety of types of software errors including modeling, design and coding errors.) The inevitability of bugs escapes very few computer users and programmers and their pervasiveness is stressed by most software, and especially safety, engineers. The dictum, "There is always another software bug,"[3] especially in the long and complex systems controlling life-critical and quality-of-life-critical technologies, captures the way in which many individuals in the business of designing, building and analyzing computer systems perceive this fact of programming life. Errors in complex functional computer systems are an inevitable presence in ambitious systems [17]. David Parnas has made a convincing case that "errors are more common, more pervasive, and more troublesome, in software than in other technologies," and that even skilled program reviewers are apt to miss flaws in programs[6].[15] Even when we factor out sheer incompetence, bugs in significant number are endemic to programming. They are the natural hazards of any substantial system.

Although this way of thinking about bugs is helpful because it underscores the vulnerability of complex systems, it also creates a problematic mind-set for

accountability. On the one hand, the standard conception of responsibility directs us to the person who either intentionally or by not taking reasonable care causes harm. On the other, the view of bugs as inevitable hazards of programming implies that while harms and inconveniences caused by bugs are regrettable, they cannot—except in cases of obvious sloppiness—be helped. In turn, this suggests that it is unreasonable to hold programmers, systems engineers, and designers, to blame for imperfections in their systems.

Parallels from other areas of technology can perhaps clarify the contrast that I am trying to draw between cases of failures for which one holds someone accountable, and frequently blameworthy, and cases where — despite the failures — one tends to hold no one accountable. As an example of the former, consider the case of the space-shuttle Challenger. Following an inquiry into the Challenger's explosion, critics found fault with NASA and Morton-Thiokol because several engineers, aware of the limitations of the O-Rings, had conveyed to management the strong possibility of failure under cold-weather-launch conditions. We hold NASA executives accountable, and judge their actions reckless, because despite this knowledge and the presence of cold-weather conditions, they went ahead with the space-shuttle launch.

In contrast, consider an experience that was common during construction of several of the great suspension bridges of the late 19th century, such as the St. Louis and Brooklyn Bridges. During construction, hundreds of bridge-workers succumbed to a mysterious disease then referred to as "the Bends", or "caisson disease"[16]. Although the working conditions and inadequate response from medical staff were responsible for the disease, we cannot assign blame for the harms suffered by the workers or find any individual or distinct group, such as the bridge companies, their chief engineers, or even their medical staff, accountable because causes and treatments of the disease were beyond the scope of medical science of the day.

For the great suspension bridges, it was necessary to sink caissons deep underground in order to set firm foundations—preferably in bedrock—for their enormous towers. Upon emerging from the caissons, workers would erratically develop an array of symptoms which might include dizziness, double vision, severe pain in torso and limbs, profuse perspiration, internal bleeding, convulsions, repeated vomiting and swollen and painful joints. For some, the symptoms would pass after a matter of hours or days, while for others symptoms persisted and they were left permanently paraplegic. Others died. While bridge doctors understood that these symptoms were related to workers' exposure to highly pressured air, they could not accurately pinpoint what caused "the bends". They offered a variety of explanations, including newness to the job, poor nutrition, and overindulgence in alcohol. They tried assigning caisson work only to those they judged to be in "prime" physical shape, reducing the time spent in the caissons, and even outfitting workers with bands of zinc and silver about their wrists, arms, and ankles. All to no avail.

We have since learned that "decompression sickness" is a condition brought on by moving too rapidly from an atmosphere of compressed air to normal atmospheric conditions. It is easily prevented by greatly slowing the rate of decompression. Ironically, a steam elevator that had been installed in both the Brooklyn Bridge and St. Louis Bridge caissons, as a means of alleviating discomfort for bridge-workers so they

would not have to make the long and arduous climb up a spiral staircase, made things all the more dangerous. Nowadays, for a project the scope of the Brooklyn Bridge, a decompression chamber would surely be provided as a means of controlling the rate of decompression. Bridge companies not following the recommended procedures would certainly be held blameworthy for harms and risks.

What is the relation of these two examples to the way we conceive of bugs? When we conceive of bugs as an inevitable byproduct of programming we are likely to judge bug-related failures in the way we judged early handling of the Bends — an inevitable, albeit unfortunate, consequence of a glorious new technology for which we hold no one accountable. The problem with this conception of bugs, is that it is a barrier to identifying cases of bug-related failure that more closely parallel the case of the Challenger. In these types of cases we see wrongdoing and expect someone to "step forward" and be answerable. The Bends case shows, too, that our standard of judgment need not remain fixed. As knowledge and understanding grows, so the standard changes. Today, bridge building companies are accountable for preventing cases of decompression sickness. An explicitly more discerning approach to bugs that indicates a range of acceptable error would better enable discrimination of the "natural hazards", the ones that are present despite great efforts and adherence to the highest standards of contemporary practice, from those that with effort and good practice, could have been avoided.

Finally, if experts in the field deny that such a distinction can be drawn, in view of the inevitability of bugs and their potential hazard, it is reasonable to think that the field of computing is not yet ready for the various uses to which it is being put.

"It's the Computer's Fault": The Computer as Scapegoat

Most of us can recall a time when someone (perhaps ourselves) offered the excuse that it was the computer's fault — the bank clerk explaining an error, the ticket agent excusing lost bookings, the student justifying a late paper. Although the practice of blaming a computer, on the face of it, appears reasonable and even felicitous, it is a barrier to accountability because, having found one explanation for an error or injury, the further role and responsibility of human agents tend to be underestimated — even sometimes ignored. As a result, no-one is called upon to answer for an error or injury.

Consider why blaming a computer *appears* plausible by applying Feinberg's analysis of blame. First, the causal condition: Computer systems frequently mediate the interactions between machines and humans, and between one human and another. This means that human actions are distanced from their causal impacts (which in some cases could be harms and injuries) and, at the same time, that the computer's action is a more direct causal antecedent. In such cases the computer satisfies the first condition for blameworthiness. Of course, causal proximity is not a sufficient condition. We do not, for example, excuse a murderer on grounds that it was the bullet entering a victim's head, and not he, who was directly responsible for the victim's death. The fault condition must be satisfied too.

Here, computers present a curious challenge and temptation. As distinct from many other inanimate objects, computers perform tasks previously performed by

humans in positions of responsibility. They calculate, decide, control, and remember. For this reason, and perhaps even more deeply rooted psychological reasons [20], people attribute to computers and not to other inanimate objects (like bullets) the array of mental properties, such as intentions, desires, thoughts, preferences, that lead us to judge human action faulty and make humans responsible for their actions[17]. Were a loan adviser to approve a loan to an applicant who subsequently defaulted on the loan, or a doctor to prescribe the wrong antibiotic for a patient who died, or an intensive care attendant incorrectly to assess the prognosis for an accident victim and deny the patient a respirator, we would hold accountable the loan adviser, the doctor, and the attendant. When these human agents are replaced with computerized counterparts (the computerized loan adviser, and expert systems MYCIN, that suggests the appropriate antibiotics for a given conditions, and APACHE, a system that predicts a patient's chance of survival [22]), it may seem reasonable to hold the systems answerable for harms. That is, there is a prima facie case in favor of associating blame with the *functions* even though they are now performed by computer systems and not humans.

Not all cases in which people blame computers rest on this tendency to attribute to computers the special characteristics that mark humans as responsible agents. In at least some cases, by blaming a computer, a person is simply shirking responsibility. In others, typically cases of collective action, a person cites a computer because she is genuinely baffled about who *is* responsible. When an airline reservation system malfunctions, for example, lines of accountability are so obscure that to the ticket agent the computer indeed is the most salient causal antecedent of the problem. Here, the computer serves as a stopgap for something elusive, the one who is, or should be, accountable. Finally, there are the perplexing cases, discussed earlier, where computers perform functions previously performed by humans in positions of responsibility leading to the illusion of computers as moral agents capable of assuming responsibility. (For interesting discussions of the viability of holding computers morally responsible for harms see [11, 23]) In the case of an expert system, working out new lines of accountability may point to designers of the system, the human experts who served as sources, or the organization that chooses to put the system to use[18]. Unless alternate lines of accountability are worked out, accountability for these important functions will be lost.

Ownership without Liability

The issue of property rights over computer software has sparked active and vociferous public debate. Should program code, algorithms, user-interface ("look-and-feel"), or any other aspects of software be privately ownable? If yes, what is the appropriate form and degree of ownership—trade secrets, patents, copyright, or a new (*sui generis*) form of ownership devised specifically for software? Should software be held in private ownership at all? Some have clamored for software patents, arguing that protecting a strong right of ownership in software, permitting owners and authors to "reap rewards," is the most just course. Others urge social policies that would place software in the public domain, while still others have sought explicitly to balance owners' rights with broader and longer-term social interests and the advancement of

computer science [25-27]. Significantly, and disappointingly, absent in these debates is any reference to owners' responsibilities[19].

While ownership implies a bundle of rights, it also implies responsibilities. In other domains, it is recognized that along with the privileges and profits of ownership comes responsibility. If a tree branch on private property falls and injures a person under it, if a pet Doberman escapes and bites a passerby, the owners are accountable. Holding owners responsible makes sense from a perspective of social welfare because owners are typically in the best position to control their property directly. Likewise in the case of software, its owners (usually the producers) are in the best position to affect the quality of the software they release to the public. Yet the trend in the software industry is to demand maximal property protection while denying, to the extent possible, accountability. This trend creates a vacuum in accountability as compared with other contexts in which a comparable vacuum would be filled by property owners.

This denial of accountability can be seen, for example, in the written license agreements that accompany almost all mass-produced consumer software which usually includes one section detailing the producers' rights, and another negating accountability. According to most versions of the license agreement, the consumer merely licenses a copy of the software application and is subject to various limitations on use and access, while the producer retains ownership over the program itself as well as the copies on floppy-disk. The disclaimers of liability are equally explicit. Consider, for example, phrases taken from the Macintosh Reference Manual (1990): "Apple makes no warranty or representation, either expressed or implied with respect to software, its quality, performance, merchantability, or fitness for a particular purpose. As a result, this software is sold 'as is,' and you, the purchaser are assuming the entire risk as to its quality and performance." The Apple disclaimer goes on to say, "In no event will Apple be liable for direct, indirect, special, incidental, or consequential damages resulting from any defect in the software or its documentation, even if advised of the possibility of such damages."

The result is that software is released in society, for which users bear the risks, while those who are in the best position to take accountability for potential harms and risks appear unwilling to do so. Although several decades ago software developers might reasonably have argued that their industry was not sufficiently well developed to be able to absorb the potentially high cost of the risks of malfunction, the evidence of present conditions suggests that a re-evaluation is well warranted. The industry has matured, is well-entrenched, reaches virtually all sectors of the economy, and quite clearly offers the possibility of stable and sizable profit. It is therefore appropriate that the industry be urged to acknowledge accountability for the burden of its impacts.

Restoring Accountability

The systematic erosion of accountability is neither a necessary nor inevitable consequence of computerization; rather it is a consequence of co-existing factors discussed above: many hands, bugs, computers-as-scapegoat, and ownership without liability, which act together to obscure accountability[20]. Barriers to accountability are

not unique to computing. Many hands create barriers to responsible action in a wide range of settings, including technologies other than computing; failures can beset other technologies even if not to the degree, and in quite the same way, as bugs in computer systems. The question of who should bear the risks of production — owners or users — is not unique to computing. Among the four, citing the computer as scapegoat may be one that is more characteristic of computing than of other technologies. The coincidence of the four barriers, perhaps unique to computing, makes accountability in a computerized society a problem of significant proportion. I conclude with the suggestion of three possible strategies for restoring accountability.

An Explicit Standard of Care – A growing literature discusses guidelines for safer and more reliable computer systems (for example, [2, 6]). Among these guidelines is a call for simpler design, a modular approach to system building, meaningful quality assurance, independent auditing, built-in redundancy, and excellent documentation. Some authors argue that better and safer systems would result if these guidelines were expressed as an explicit standard of care taken seriously by the computing profession, promulgated through educational institutions, urged by professional organizations, and even enforced through licensing or accreditation[21]. Naturally, this would not be a fixed standard but one that evolved along with the field. What interests me here, however, is another potential payoff of an explicit standard of care; namely, a non-arbitrary means of determining accountability. A standard of care offers a way to distinguish between malfunctions (bugs) that are the result of inadequate practices, and the failures that occur in spite of a programmer's or designer's best efforts, for distinguishing analogs of the failure to alleviate the Bends in 19th-century bridge workers, from analogs of the Challenger space-shuttle. Had the guidelines discussed by Leveson and Turner[3], for example, been accepted as a standard of care at the time the Therac-25 was created, we would have had the means to establish that corporate developers of the system were accountable for the injuries. As measured against these guidelines they were negligent and blameworthy.

By providing an explicit measure of excellence that functions independently of pressures imposed by an organizational hierarchy within which some computer systems engineers in corporations and other large organizations are employed, a standard of care could also function to back up professional judgment. It serves to bolster an engineer's concern for safety where this concern conflicts with, for example, institutional frugality. A standard of care may also be a useful vehicle for assessing the integrity of the field of computing more broadly. In a point raised earlier, I suggested that it is important to have a good sense of whether or when the "best efforts" as recognized by a field — especially one as widely applied as computing — are good enough for the many uses to which they are put.

Distinguishing Accountability from Liability – For many situations in which issues of responsibility arise, accountability and liability are strongly linked. In spite of their frequent connection, however, their conceptual underpinnings are sufficiently distinct so as to make a difference in a number of important contexts. One key difference is that appraisals of liability are grounded in the plight of a victim, whereas appraisals of

accountability are grounded in the relationship of an agent to an outcome.[22] The starting point for assessing liability is the victim's condition; liability is assessed backward from there. The extent of liability, frequently calculated in terms of sums of money, is determined by the degree of injury and damage sustained by any victims. The starting point for assessing accountability is the nature of an action and the relationship of the agent (or several agents) to the action's outcome. (In many instances, accountability is mediated through conditions of blameworthiness, where the so-called "causal" and "fault" conditions would be fulfilled.) Although those people who are accountable for a harm are very frequently the same as those who are liable, merging the notions of liability with accountability, or accepting the former as a substitute for the latter, can obscure accountability in many of the contexts targeted in earlier sections of this paper. Consider, for example, the problem of many hands and how it is affected by this.

The problem of many hands is profound and seems unlikely to yield easily to a general, or slick, solution. For the present, a careful case-by-case analysis of a given situation in order to identify relevant causal factors and fault holds the most promise. Such analysis is rarely easy or obvious for the much studied, widely publicized catastrophes such as the Therac-25, or the Challenger, and perhaps even more so for the preponderant smaller scale situations in which accountability is nevertheless crucial. Our grasp of accountability can be obscured, however, if we fail to distinguish between accountability and liability. Consider why. In cases of collective (as opposed to individual) action, if all we care about is liability, it makes sense to share the burden of compensation among the collective in order to lighten the burden of each individual. Moreover, because compensation is victim-centered, targeting one satisfactory source of compensation (the so-called "deep pocket"), can and often does let others "off the hook." In contrast, where we care about accountability, many hands do not offer a means of lessening or escaping its burden. No matter how many agents there are, each may be held equally and fully answerable to a given harm[23]. There is no straightforward analog with the deep-pocket phenomenon.

Although a good system of liability offers a *partial* solution because at least the needs of victims are addressed, it can deflect attention away from accountability. Decision-makers may focus exclusively on liability and fail to grasp the extent of their answerability for actions and projects they plan. The Ford-Pinto case provides an example. Although the case as a whole is too complex to be summarized in a few sentences one aspect bears directly on this issue. According to various reports, when Ford executives considered various options for the design of the Pinto, they focused on liability and predicted that losses due to injury-liability lawsuits for the cheaper design would be offset by the expected savings[24]. Ford corporation could spread the anticipated losses so as not to be significantly affected by them. By spreading the liability thin enough and covering it by the savings from the cheaper design, no individual or part of the company would face a cost too heavy to bear.

If Ford executives had been thinking as carefully about answerability (which cannot be spread, thinned and offset) as they were about liability, their decision might well have been different. I do not hereby impugn the general method of cost-benefit analysis for business decisions of this sort. Rather, I suggest that in reckoning only

with liability, the spectrum of values the executives considered was too narrow and pushed them in the wrong direction. A professional culture where accountability prevails, where the possibility exists for each to be called to answer for his or her decisions, would not as readily yield to decisions like the one made by Ford executives. Many hands need not make, metaphorically speaking, the burden lighter.

Strict Liability and Producer Responsibility – In the previous section I suggested that liability should not be understood as a substitute for accountability. Acknowledging, or for that matter denying, one's liability for an outcome does not take care of one's answerability for it. Nevertheless, establishing adequate policies governing liability for impacts of computerization is a powerful means of expressing societal expectations and at least partially explicates lines of accountability. Well-articulated policies on liability would serve the practical purpose of protecting public interests against some of the risks of computer system failure which are further amplified by a reluctance on the part of producers and owners of systems-in-use to be accountable for them. I propose that serious consideration be given to a policy of strict liability for computer system failure, in particular for those sold as consumer products in mass markets.

To be strictly liable for a harm is to be liable to compensate for it even though one did not bring it about through faulty action. (In other words, one "pays for" the harm if the causal condition is satisfied even though the fault condition is not.) This form of liability, which is found in the legal codes of most countries, is applied, typically, to the producers of mass-produced consumer goods, potentially harmful goods, and to the owners of "ultra-hazardous" property. For example, milk producers are strictly liable for illness caused by spoiled milk, even if they have taken a normal degree of care; owners of dangerous animals (for example, tigers in a circus) are strictly liable for injuries caused by escaped animals even if they have taken reasonable precautions to restrain them.

Supporters of strict liability argue that it is justified, in general, because it benefits society by placing the burden of risk where it best belongs. Its service to the public interest is threefold. First, it protects society from the risks of potentially harmful or hazardous goods and property by providing an incentive to sellers of consumer products and owners of potentially hazardous property to take *extra*ordinary care. Second, it seeks compensation for victims from those best able to afford it, and to guard against the harm. And third, it reduces the cost of litigation by eliminating the onerous task of proving fault. Critics, on the other hand, argue that not only is strict liability unjust, because people are made to pay for harms that were not their fault, but it might indeed work *against* the public interest by discouraging innovative products. Because of the prohibitive cost of bearing the full risk of malfunction and injury, many an innovation might not be pursued through fear of ruin. In the case of new and promising, but not yet well-established technologies, this argument may hold even more sway.

Whether or not strict liability is a good general strategy is an issue best reserved for another forum. However, themes from the general debate can cast light on its merits or weaknesses as a response to computer system failure, especially since our present system of liability does include strict liability as a viable answer. In early days

of computer development, recognition of both the fragility and the promise of the field might have argued for an extra degree of protection for producers by allowing risk to be shifted to consumers and other users of computing. In other words, those involved in the innovative and promising developments were spared the burden of liability. Over the course of several decades we have witnessed a maturing of the field, which now shows clear evidence of strength and vitality. The argument for special protection is therefore less compelling. Furthermore, computing covers a vast array of applications, many resembling mass-produced consumer goods, and a number that are life-critical. This argues for viewing producers of computer software in a similar light to other producers of mass-produced consumer goods and potentially harm-inducing products.

By shifting the burden-of-accountability to the producers of defective software, strict liability would also address a peculiar anomaly. One of the virtues of strict liability is that it offers a means of protecting the public against the potential harms of risky artifacts and property. Yet in the case of computing and its applications, we appear to live with a strange paradox. On the one hand, the prevailing lore portrays computer software as prone to error in a degree surpassing most other technologies, and portrays bugs as an inevitable by-product of computing itself. Yet on the other hand, most producers of software explicitly deny accountability for the harmful impacts of their products, even when they malfunction. Quite the contrary should be the case. Because of the always-lurking possibility of bugs, software seems to be precisely the type of artifact for which strict liability is appropriate; it would assure compensation for victims, and send an emphatic message to producers of software to take *extra*-ordinary care to produce safe and reliable systems.

NOTES

1 Several people have contributed generously to this work. Michael Davis, Deborah G. Johnson, Arthur Kuflik, Pamela Samuelson, Debra Satz, Richard De George, Larry May, and Dennis Thompson read drafts and made invaluable suggestions. Reviewers for *Science and Engineering Ethics* offered thorough and challenging commentary. An earlier version of the paper was presented at The American Philosophical Association, Eastern Division Meeting, December 1993, where audience comments and questions led to clarification of several key issues.

2 G.R. Driver and John C. Miles (Editors and Translators), *The Babylonian Laws*, Volume II, Clarendon Press, Oxford, 1955.

3 For example, Joseph Weizenbaum, and more recently Nancy Leveson, Peter Neumann, David Parnas, and others.

4 The community of people who dedicate a significant proportion of their time and energy to building computer and computerized systems, and to those engaged in the science, engineering, design, and documentation of computing.

5 Thanks to Deborah Johnson for suggesting this phrase.

6 Of course there are many situations in which harm and injury occur but are no one's fault; that is, no one is to blame for them.

7 Feinberg's analysis is more complex, involving several additional conditions and refinements. Since these are not directly relevant to our discussion, for the sake of simplicity I have omitted them here.

8 Readers interested in this case may refer to Denning P (1990) *Computers Under Attack*. ACM Press, New York.

9 The overlap, though significant, is only partial. Take for example, circumstances in which a person — frequently an office-holder — is by convention, or by prior agreement, held answerable. In such cases one would expect accountability even though blame is not appropriate.

10 This phrase was first coined by Dennis Thompson in his book-chapter "The Moral Responsibility of Many Hands"[8] which discusses the moral responsibilities of political office holders and public officials working within large government bureaucracies.

11 Most users of personal computers will have experienced occasions when their computers freeze. Neither the manufacturer of the operating system nor of the applications assume responsibility for this, preferring to blame the problem on "incompatibilities".

12 The primary sources for my discussion are Leveson and Turner's excellent and detailed account[3] and an earlier paper by Jacky[16].

13 Much credit is due to Fritz Hager, the hospital physicist in Tyler, Texas, who took upon himself the task of uncovering the problem and helped uncover software flaws.

14 Michael Davis, in commenting on this paper, points out that in certain contexts, for example automobile accidents, our manner of speaking allows for "accidents" for which we may yet blame someone.

15 See also [18] for an explanation of why software is particularly prone to errors.

16 This case is drawn from David McCullough's book about the building of the Brooklyn Bridge[19].

17 Apparently this phenomenon is firmly rooted. In a study by Friedman and Millett, interviews with male undergraduate computer science majors found that a majority attributed aspects of agency to computers and significant numbers held computers morally responsible for errors[21].

18 B. Friedman and P. Kahn argue that systems designers play an important role in preventing the illusion of the computer as a moral agent. They argue that certain prevalent design features, such as anthropomorphizing a system, delegating decision making to it, and delegating instruction to it diminish a user's sense of agency and responsibility[24].

19 For an exception see Samuelson's recent discussion of liability for defective information[28].

20 Here and elsewhere I should not be understood as suggesting that the four barriers give a complete explanation of failures in accountability.

21 The issue of licensing software producers remains controversial.

22 Dennis Thompson points out that common usage of the two terms may not track this distinction as precisely as I suggest. For purposes of this discussion I hope to hold the issue of terminology at bay and focus on the underlying ideas and their relevant distinctiveness.

23 Compare this to the judge's finding in the "Red Hook Murder."[29] Even though it was almost certainly known which one of the three accused pulled the trigger, the court viewed all three defendants to be equal and "deadly conspirators" in the death of the victim Patrick Daley.

24 This prediction turned out, in fact, to be inaccurate, but this is not relevant to our central concern.

REFERENCES

[1] Borning A (1987) Computer System Reliability and Nuclear War. *Communications of the ACM* **30, 2**: 112-131.

[2] Leveson N (1986) Software Safety: Why, What, and How. *Computing Surveys* **18, 2**: 125-163.

[3] Leveson N, and Turner C (1993) An Investigation of the Therac-25 Accidents. *Computer* **26, 7**: 18-41.

[4] Littlewood B and Strigini L (1992) The Risks of Software. *Scientific American,* November: 62-75.

[5] Neumann P G. (monthly column) Inside Risks. *Communications of the ACM.*

[6] Parnas D, Schouwen J and Kwan S P (1990) Evaluation of Safety-Critical Software. *Communications of the ACM* **33, 6**: 636-648.
[7] Feinberg, J (1985) Sua Culpa, In: Johnson D G and Snapper J, eds. *Ethical Issues in the Use of Computers*. Wadsworth, Belmont.
[8] Thompson D (1987) *Political Ethics and Public Office*. Harvard University Press, Cambridge.
[9] De George R (1991) Ethical Responsibilities of Engineers in Large Organizations: The Pinto Case, In: May L and Hoffman S, eds. *Collective Responsibility*. Rowman and Littlefield: 151-166.
[10] Feinberg J. (1970) Collective Responsibility. In Feinberg J, ed. *Doing and Deserving*, Princeton University Press, Princeton.
[11] Ladd J. (1989) Computers and Moral Responsibility: A Framework for an Ethical Analysis. In Gould C, ed. *The Information Web: Ethical and Social Implications of Computer Networking*. Westview Press, Boulder.
[12] Thompson D (1987) The Moral Responsibility of Many Hands. In: *Political Ethics and Public Office*, Harvard University Press, Cambridge: 46-60.
[13] Velasquez M (1991) Why Corporations Are Not Morally Responsible for Anything They Do. In: May L and Hoffman S, eds. *Collective Responsibility*. Rowman and Littlefield: 111-131.
[14] Johnson D G and Mulvey J M (1993) *Computer Decisions: Ethical Issues of Responsibility and Bias*. Statistics and Operations Research Series, Princeton University, SOR-93-11.
[15] Weizenbaum J (1972) On the Impact of the Computer on Society. *Science* **176, 12**: 609-614.
[16] Jacky J (1989) *Safety-Critical Computing: Hazards, Practices, Standards and Regulations* (Unpublished Manuscript). University of Washington.
[17] Corbato F J (1991) On Building Systems That Will Fail. *Communications of the ACM* **34, 9**: 73-81.
[18] Smith, B C (1985) *The Limits of Correctness*. Center for the Study of Language and Information, Stanford University, CSLI-85-35.
[19] McCullough D (1972) *The Great Bridge*. Simon & Schuster, New York.
[20] Turkle, Sherry (1984) *The Second Self*. Simon & Schuster, New York.
[21] Friedman B and Millett L (1995) "It's the computer's fault—reasoning about computers as moral agents." Paper presented at CHI '95, New York.
[22] Fitzgerald S (1992) Hospital Computer Predicts Patients' Chance of Survival. *The Miami Herald*, Sunday, July 19.
[23] Snapper J W (1985) Responsibility for Computer-Based Errors. *Metaphilosophy* **16**: 289-295.
[24] Friedman B and Kahn P H (1992) Human Agency and Responsible Computing: Implications for Computer System Design. *Journal of System Software* **17**, January: 7-14.
[25] Nissenbaum H. Should I Copy My Neighbor's Software? In: Johnson D G and Nissenbaum H (1995) *Computers, Ethics, and Social Values*. Prentice-Hall, Englewood.
[26] Samuelson P (1992) *Adapting Intellectual Property Law to New Technologies: A Case Study on Computer Programs*. National Research Council Report.
[27] Stallman R M (1987) The GNU Manifesto. *GNU Emacs Manual*: 175-84.
[28] Samuelson P (1993) Liability for Defective Information. *Communications of the ACM* **36, 1**: 21-26.
[29] Fried J P (1993) Maximum Terms for Two Youths in Red Hook Murder. *New York Times*, July 7: 3.