UNIQUE ETHICAL PROBLEMS IN INFORMATION TECHNOLOGY
By Walter Maner

Department of Computer Science
Bowling Green State University
Bowling Green, OH 43403
USA

maner@cs.bgsu.edu
http://web.cs.bgsu.edu/maner

This paper appeared in *Science and Engineering Ethics*, volume 2, number 2 (April, 1996), pages 137-154.

ABSTRACT

A distinction is made between moral indoctrination and instruction in ethics. It is argued that the legitimate and important field of computer ethics should not be permitted to become mere moral indoctirnation. Computer ethics is an academic field in its own right with unique ethical issues that would not have existed if computer technology had not been invented. Several example issues are presented to illustrate this point. The failure to find satisfactory non-computer analogies testifies to the uniqueness of computer ethics. Lack of an effective analogy forces us to discover new moral values, formulate new moral principles, develop new policies, and find new ways to think about the issues presented to us. For all of these reasons, the kind of issues presented deserves to be addressed separately from others that might at first appear similar. At the very least, they have been so transformed by computing technology that their altered form demands special attention.

INTRODUCTION

One factor behind the rise of computer ethics is the lingering suspicion that computer professionals may be unprepared to deal effectively with the ethical issues that arise in their workplace. Over the years, this suspicion has been reinforced by mostly anecdotal research that seems to show that computer professionals simply do not recognize when ethical issues are present. Perhaps the earliest work of this kind was done by Donn Parker in the late 1970s at SRI International.[1]

In 1977, Parker invited highly trained professionals from various fields to evaluate the ethical content of 47 simple hypothetical cases that he had created based in part on his expert knowledge of computer abuse. Workshop participants focused on each action or non-action of each person who played a role in these one-page scenarios. For each act that was performed or not performed, their set task was to determine whether the behavior was unethical or not, or simply raised no ethics issue at all. Parker found a surprising amount of residual disagreement among these professionals even after an exhaustive analysis and discussion of all the issues each case presented.

More surprisingly, a significant minority of professionals held to their belief that no ethics issue was present even in cases of apparent computer abuse. For example, in Scenario 3.1, a company representative routinely receives copies of the computerized arrest records for new company employees. These records are provided as a favor by a police file clerk who happens to have access to various local and federal databases containing criminal justice information. Nine of the 33 individuals who analyzed this case thought disclosure of arrest histories raised no ethics issues at all. Parker's research does not identify the professions represented by those who failed to detect ethics issues, but most of the participants in this early study[2] were computer professionals. This left casual readers of Parker's *Ethical Conflicts in Computer Science and Technology* free to identify computer professionals as the ones who lacked ethical sensitivity. If some of them could not even recognize when ethical issues were present, it is hard to imagine how they could ever hope to deal responsibly with them. According to Parker, the problem may have been fostered by computer education and training programs that encouraged, or at least failed to criminalize, certain types of unethical professional conduct.[3]

This perception of professional inadequacy is part of a largely hidden political agenda that has contributed to the development of various curricula in computer ethics. In recent years, the tacit perception that those preparing for careers in computing may need remedial moral education seems to have influenced some accreditation boards. As a result, they have been willing to mandate more and more ethical content in computer science and computer engineering programs.

They may also be responding to the increased media attention given to instances of computer abuse, fraud and crime. Others demand more ethical content because they believe that catastrophic failures of computer programs are directly attributable to immoral behavior.[4]

The growth of interest is gratifying, especially considering that, in 1976, I found it hard to convince anyone that "computer ethics" was anything other than an oxymoron.[5] No doubt Norbert Weiner would be pleased to see his work bearing late fruit.[6] At the same time, I am greatly disturbed when courses in social impact and computer ethics become a tool for indoctrination in appropriate standards of professional conduct. Donald Gotterbarn, for example, argues that one of the six goals of computer ethics is the "socialization" of students into "professional norms."[7] The fact that these norms are often eminently reasonable, even recommended thoughtfully to us by our professional organizations, does not make indoctrination any less repugnant. The goal cannot be simply to criminalize or stigmatize departures from professional norms. Consider an analogy. Suppose a course in Human Sexual Relationships has for its goal the socialization of college students into "high standards" of sexual conduct, and that this goal is enforced by contradicting or discrediting anyone who violates these standards. Most people would be quick to recognize that this curriculum is more political than academic, and that such an approach would tend to create a classroom environment where bias could overwhelm inquiry.

We stand today on the threshold of a time when well-intended political motives threaten to reshape computer ethics into some form of moral education. Unfortunately, it is an easy transition from the *correct* belief that we ought to teach future computer scientists and engineers the meaning of responsible conduct, to the *mistaken* belief that we ought to train them to behave like responsible professionals. When Terrell Bynum says, for example, that he hopes the study of computer ethics will develop "good judgment" in students,[8] he is not advocating socialization. By "good judgment" he means to refer to the reasoned and principled process by which reflective moral judgments are rendered. From this correct position, it is a tempting and subtle transition to the mistaken position that computer ethics should cause students to develop good *judgments,* meaning that their positions on particular moral issues conform to the norms of the profession. This self-deceiving mistake occurs because there is an undetected shift in emphasis from the *process* to the *products* of moral deliberation.

My point is that a perceived need for moral education does not and cannot provide an adequate rationale for the study of computer ethics. Rather, it must exist as a field worthy of study in its own right and not because at the moment it can provide useful means to certain socially noble ends. To exist and to *endure* as a separate field, there must be a unique domain for computer ethics distinct from the domain for moral education, distinct even from the domains of other kinds of professional and applied ethics. Like James Moor, I believe computers are special technology and raise special ethical issues,[9] hence that computer ethics deserves special status.

My remaining remarks will suggest a rationale for computer ethics based on arguments and examples showing that one of the following is true:

- that certain ethical issues are so transformed by the use of computers that they deserve to be studied on their own, in their radically altered form,

or

- that the involvement of computers in human conduct can create entirely new ethical issues, unique to computing, that do not surface in other areas.

I shall refer to the first as the "weaker view" and the second as the "stronger view." Although the weaker view provides sufficient rationale, most of my attention will be focused on establishing the stronger view. This is similar to the position I took in 1980[10] and 1985[11], except that I no longer believe that problems merely aggravated by computer technology deserve special status.

LEVELS OF JUSTIFICATION FOR THE STUDY OF COMPUTER ETHICS

From weaker to stronger, there are at least six levels of justification for the study of computer ethics.

Level One

We should study computer ethics because doing so will make us behave like responsible professionals.

At worst, this type of rationale is a disguised call for moral indoctrination. At best, it is weakened by the need to rely on an elusive connection between right knowledge and right conduct. This is similar to the claim that we should study religion because that will cause us to become more spiritual. For some people, perhaps it may, but the mechanism is not reliable.

Level Two

We should study computer ethics because doing so will teach us how to avoid computer abuse and catastrophes.

Reports by Parker,[12] Neumann,[13] Forester and Morrison[14] leave little doubt that computer use has led to significant abuse, hijinks, crime, near catastrophes, and actual catastrophes. The question is: Do we get a balanced view of social responsibility merely by examining the profession's dirty laundry? Granted, a litany of computer "horror stories" does provide a vehicle for infusing some ethical content into the study of computer science and computer engineering. Granted, we should all work to prevent computer catastrophes. Even so, there are major problems with the use of conceptual shock therapy:

- The cases commonly used raise issues of bad conduct rather than good conduct. They tell us what behaviors to avoid but do not tell us what behaviors are worth modeling.

- As Leon Tabak has argued, this approach may harm students by preventing them from developing a healthy, positive and constructive view of their profession.[15]

- Most horror stories are admittedly rare and extreme cases, which makes them seem correspondingly remote and irrelevant to daily professional life.

- Persons who use computers for abusive purposes are likely to be morally bankrupt. There is little we can learn from them.

- Many computer catastrophes are the result of *unintended* actions and, as such, offer little guidance in organizing *purposive* behavior.

- A litany of horror stories does not itself provide a coherent concept of computer ethics.

Level Three

We should study computer ethics because the advance of computing technology will continue to create temporary policy vacuums.

Long-term use of poorly designed computer keyboards, for example, exposes clerical workers to painful, chronic, and eventually debilitating repetitive stress injury. Clearly employers should not require workers to use equipment that will likely cause them serious injury. The question is: What policies should we formulate to address problems of long-term keyboard use? New telephone technology for automatic caller identification creates a similar policy vacuum. It is not immediately obvious what the telephone company should be required to do, if anything, to protect the privacy of callers who wish to remain anonymous.

Unlike the first- and second-level justifications I have considered and rejected, this third-level justification does appear to be sufficient to establish computer ethics as an important and independent discipline. Still, there are problems:

- Since policy vacuums are temporary and computer technologies evolve rapidly, anyone who studies computer ethics would have the perpetual task of tracking a fast-moving and ever-changing target.

- It is also possible that practical ethical issues arise mainly when policy frameworks clash. We could not resolve such issues merely by formulating more policy.

Level Four

We should study computer ethics because the use of computing permanently transforms certain ethical issues to the degree that their alterations require independent study.

I would argue, for example, that many of the issues surrounding intellectual property have been radically and permanently altered by the intrusion of computer technology. The simple question, “What do I own?” has been transformed into the question, “What exactly is it that I own when I own something?” Likewise, the availability of cheap, fast, painless, transparent encryption technology has completely transformed the privacy debate. In the past, we worried about the erosion of privacy. Now we worry about the impenetrable wall of computer-generated privacy afforded to every criminal with a computer and half a brain.

Level Five

We should study computer ethics because the use of computing technology creates, and will continue to create, novel ethical issues that require special study. I will return to this topic in a moment.

Level Six

We should study computer ethics because the set of novel and transformed issues is large enough and coherent enough to define a new field. I mention this hopefully as a theoretical possibility. Frankly, after fifteen years, we have not been able to assemble a critical mass of self-defining core issues. Joseph Behar, a sociologist, finds computer ethics diffuse and unfocused.[16] Gary Chapman, when he spoke to the Computers and Quality of Life Conference in 1990, complained that no advances had been made in computer ethics.[17] There are various explanations for this apparent (or real) lack of progress:[18]

- Computer ethics is barely fifteen years old.[19] Much of its intellectual geography remains uncharted.

- So far, no one has provided a complete and coherent concept of the proper subject matter for computer ethics.

- We have wrongly included in the domain of computer ethics any unethical act that happened to involve a computer. In the future, we must be more careful to restrict ourselves to those few acts where computers have an *essential* as opposed to *incidental* involvement.

- Because computer ethics is tied to an evolving technology, the field changes whenever the technology changes. For example, the use of networked computers presents moral problems different from those presented by the use of standalone computers. The use of mouse-driven interfaces raises issues different from those raised by keyboard-driven interfaces, particularly for people who are blind.

- We adopted, from clever philosophers, the dubious practice of using highly contrived, two-sided, dilemmatic cases to expose interesting but irresolvable ethical conflicts. This led to the false perception that there could be no progress and no commonality in computer ethics. New research may cause this perception to fade.[20]

- We have remained focused for too long on the dirty laundry of our profession.

On a hopeful note, the ImpactCS Steering Committee chaired by C. Dianne Martin is halfway through a three-year NSF-funded project that will likely generate a highly coherent picture of how the computer science curriculum can address social and ethical issues. ImpactCS intends to publish specific curriculum guidelines along with concrete models for implementing them.[21]

THE SPECIAL STATUS OF COMPUTER ETHICS

I now turn to the task of justifying computer ethics at Level 5 by establishing, through several examples, that there are issues and problems unique to the field.

It is necessary to begin with a few disclaimers. First, I do not claim that this set of examples is in any sense complete or representative. I do not even claim that the kinds of examples I will use are the best kind of examples to use in computer ethics. I do not claim that any of these issues is central to computer ethics. Nor am I suggesting that computer ethics should be limited to just those issues and problems that are unique to the field. I merely want to claim that each example is, in a specific sense, unique to computer ethics.

By “unique” I mean to refer to those ethical issues and problems that

- are characterized by the primary and essential involvement of computer technology,
- exploit some unique property of that technology, and
- would not have arisen without the essential involvement of computing technology

I mean to allow room to make either a strong or a weak claim as appropriate. For some examples, I make the strong claim that the issue or problem would not have arisen *at all*. For other examples, I claim only that the issue or problem would not have arisen *in its present, highly altered form*.

To establish the essential involvement of computing technology, I will argue that these issues and problems have no satisfactory ***non***-computer moral analog. For my purposes, a "satisfactory" analogy is one that (a) is based on the use of a machine other than a computing machine and (b) allows the ready transfer of moral intuitions from the analog case to the case in question. In broad strokes, my line of argument will be that certain issues and problems are unique to computer ethics because they raise ethical questions that depend on some unique property of prevailing computer technology. My remarks are meant to apply to discrete-state stored-program inter-networking fixed-instruction-set serial machines of von Neumann architecture. It is possible that other designs (such as the Connection Machine) would exhibit a different set of unique properties.

Next I offer a series of examples, starting with a simple case that allows me to illustrate my general approach.

EXAMPLE 1: Uniquely Stored

One of the unique properties of computers is that they must store integers in "words" of a fixed size. Because of this restriction, the largest integer that can be stored in a 16-bit computer word is 32,767. If we insist on an exact representation of a number larger than this, an "overflow" will occur with the result that the value stored in the word becomes corrupted. This can produce interesting and harmful consequences. For example, a hospital computer system in Washington, D.C., broke down on September 19, 1989, because its calendar calculations counted the days elapsed since January 1, 1900. On the 19th of September, exactly 32,768 days had elapsed, overflowing the 16-bit word used to store the counter, resulting in a collapse of the entire system and forcing a lengthy period of manual operation.[22] At the Bank of New York, a similar 16-bit counter overflowed, resulting in a $32 billion overdraft. The bank had to borrow $24 million for one day to cover the overdraft. The interest on this one-day loan cost the bank about $5 million. In addition, while technicians attempted to diagnose the source of the problem, customers experienced costly delays in their financial transactions.[23]

Does this case have a satisfactory non-computer analog? Consider mechanical adding machines. Clearly they are susceptible to overflow, so it is likely that accountants who relied on them in years past sometimes produced totals too large for the machine to store. The storage mechanism overflowed, producing in steel the same result that the computer produced in silicon. The problem with this “analogy” is that, in a broad and relevant sense, adding machines *are* computers, albeit of a primitive kind. The low-level *logical* descriptions of adding machines and computers are fundamentally identical.

Perhaps your automobile’s mechanical odometer gauge provides a better analogy. When the odometer reading exceeds a designed-in limit, say 99,999.9 miles, the gauge overflows and returns to all zeros. Those who sell used cars have taken unfair advantage of this property. They use a small motor to overflow the gauge manually, with the result that the buyer is unaware that he or she is purchasing a high-mileage vehicle.

This does provide a non-computer analogy, but is it a satisfactory analogy? Does it allow the ready transfer of moral intuitions to cases involving word overflow in computers? I believe it falls short. Perhaps it would be a satisfactory analogy if, when the odometer overflowed, the engine, the brakes, the wheels, and every other part of the automobile stopped working. This does not in fact happen because the odometer is not highly coupled to other systems critical to the operation of the vehicle. What is different about computer words is that they are deeply embedded in highly integrated subsystems such that the corruption of a single word threatens to bring down the operation of the entire computer. What we require, but do not have, is a non-computer analog that has a similar catastrophic failure mode.

So the incidents at the hospital in Washington, D.C., and the Bank of New York meet my three basic requirements for a unique issue or problem. They are characterized by the primary and essential involvement of computer technology, they depend on some unique property of that technology, and they would not have arisen without the essential involvement of computing technology. Even if the mechanical adding machine deserves to be considered as an analog case, it is still true that computing technology has radically altered the form and scope of the problem. On the other hand, if the adding machine does *not* provide a good analogy, then we may be entitled to a stronger conclusion: that these problems would not have arisen *at all* if there were no computers in the world.

EXAMPLE 2: Uniquely Malleable

Another unique characteristic of computing machines is that they are very general-purpose machines. As James Moor observed, they are "logically malleable" in the sense that "they can be shaped and molded to do any activity that can be characterized in terms of inputs, outputs, and connecting logical operations."[24] The unique adaptability and versatility of computers have important moral implications. To show how this comes about, I would like to repeat a story first told by Peter Green and Alan Brightman.

Alan (nickname "Stats") Groverman is a sports fanatic and a data-crunching genius.

> His teachers describe him as having a "head for numbers." To Stats, though, it's just what he does; keeping track, for example of yards gained by each running back on his beloved [San Francisco] 49ers team. And then averaging those numbers into the season's statistics. All done in his head-for-numbers. All without even a scrap of paper in front of him.
>
> Not that paper would make much of a difference. Stats has never been able to move a finger, let alone hold a pencil or pen. And he's never been able to press the keys of a calculator. Quadriplegia made these kinds of simplicities impossible from the day he was born. That's when he began to strengthen his head.
>
> Now, he figures, his head could use a little help. With his craving for sports ever-widening, his mental playing field is becoming increasingly harder to negotiate.
>
> Stats knows he needs a personal computer, what he calls "cleats for the mind." He also knows that he needs to be able to operate that computer without being able to move anything below his neck.[25]

Since computers do not care how they get their inputs, Stats ought to be able to use a head-pointer or a mouth-stick to operate the keyboard. If mouse input is required, he could use a head-controlled mouse along with a sip-and-puff tube. To make this possible, we would need to load a new device driver to modify the behavior of the operating system. If Stats has trouble with repeating keys, we would need to make another small change to the operating system, one that disables the keyboard repeat feature. If keyboard or mouse input proves too tedious for him, we could add a speech processing chip, a microphone and voice-recognition software. We have a clear duty to provide computer access solutions in cases like this, but what makes this duty so reasonable and compelling is the fact that computers are so easily adapted to user requirements.

Does there exist any other machine that forces an analogous obligation on us to assist people with disabilities? I do not believe so. The situation would be different, for example, if Stats wanted to ride a bicycle. While it is true that bicycles have numerous adjustments to accommodate the varying geometry of different riders, they are infinitely less adaptable than computers. For one thing, bicycles cannot be programmed, and they do not have operating systems. My point is that our obligation to provide universal accessibility to computer technology would not have arisen if computers were not universally adaptable. The generality of the obligation is in proportion to the generality of the machine.

While it is clear that we should endeavor to adapt other machinery -- elevators, for example -- for use by people with disabilities, the moral intuitions we have about adapting elevators do not transfer readily to computers. Differences of scale block the transfer. Elevators can only do elevator-like things, but computers can do anything we can describe in terms of input, process, and output. Even if elevators did provide a comparable case, it would still be true that the availability of a totally malleable machine so transforms our obligations that this transformation itself deserves special study.

EXAMPLE 3: Uniquely Complex

Another unique property of computer technology is its superhuman complexity. It is true that humans program computing machines, so in that sense we are masters of the machine. The problem is that our programming tools allow us to create discrete functions of arbitrary complexity. In many cases, the result is a program whose total behavior cannot be described by any compact function.[26] Buggy programs in particular are notorious for evading compact description! The fact is we routinely produce programs whose behavior defies inspection, defies understanding --programs that surprise, delight, entertain, frustrate and ultimately confound us. Even when we understand program code in its static form, it does not follow that we understand how the program works when it executes.

James Moor provides a case in point:

> An interesting example of such a complex calculation occurred in 1976 when a computer worked on the four color conjecture. The four color problem, a puzzle mathematicians have worked on for over a century, is to show that a map can be colored with at most four colors so that no adjacent areas have the same color. Mathematicians at the University of Illinois broke the problem down into thousands of cases and programmed computers to consider them. After more than a thousand hours of computer time on various computers, the four color conjecture was proved correct. What is interesting about this mathematical proof, compared to traditional proofs, is that it is largely invisible. The general structure of the proof is known and found in the program, and any particular part of the computer's activity can be examined, but practically speaking the calculations are too enormous for humans to examine them all.[27]

It is sobering to consider how much we rely on a technology we strain and stretch to understand. In the UK, for example, Nuclear Electric decided to rely heavily on computers as its primary protection system for its first nuclear-power plant, Sizewell B. The company hoped to reduce the risk of nuclear catastrophe by eliminating as many sources of human error as possible. So Nuclear Electric installed a software system of amazing complexity, consisting of 300-400 microprocessors controlled by program modules that contained more than 100,000 lines of code.[28]

It is true that airplanes, as they existed before computers, were complex and that they presented behaviors that were difficult to understand. But aeronautical engineers do understand how airplanes work because airplanes are constructed according to known principles of physics. There are mathematical functions describing such forces as thrust and lift, and these forces behave according to physical laws. *There are no corresponding laws governing the construction of computer software.*

This lack of governing law is unique among all the machines that we commonly use, and this deficiency creates unique obligations. Specifically, it places special responsibilities on software engineers for the thorough testing and validation of program behavior. There is, I would argue, a *moral* imperative to discover better testing methodologies and better mechanisms for proving programs correct. It is hard to overstate the enormity of this challenge. Testing a simple input routine that accepts a 20-character name, a 20-character address, and a 10-digit phone number would require approximately 10^{66} test cases to exhaust all possibilities. If Noah had been a software engineer and had started testing this routine the moment he stepped off the ark, he would be less than one percent finished today *even if he managed to run a trillion test cases every second*.[29] In practice, software engineers test a few boundary values and, for all the others, they use values believed to be representative of various equivalence sets defined on the domain.

EXAMPLE 4: Uniquely Fast

On Thursday, September 11, 1986, the Dow Jones industrial average dropped 86.61 points, to 1792.89, on a record volume of 237.6 million shares. On the following day, the Dow fell 34.17 additional points on a volume of 240.5 million shares. Three months later, an article appearing in *Discover* magazine asked: Did computers make stock prices plummet? According to the article,

> ... many analysts believe that the drop was accelerated (though not initiated) by computer-assisted arbitrage. Arbitrageurs capitalize on what's known as the spread: a short-term difference between the price of stock futures, which are contracts to buy stocks at a set time and price, and that of the underlying stocks. The arbitrageurs' computers constantly monitor the spread and let them know when it's large enough so that they can transfer their holdings from stocks to stock futures or vice-versa, and make a profit that more than covers the cost of the transaction. ... With computers, arbitrageurs are constantly aware of where a profit can be made. However, throngs of arbitrageurs working with the latest information can set up perturbations in the market. Because arbitrageurs are all "massaging" the same basic information, a profitable spread is likely to show up on many of their computers at once. And since arbitrageurs take advantage of small spreads, they must deal in great volume to make it

> worth their while. All this adds up to a lot of trading in a little time, which can markedly alter the price of a stock.[30]

After a while, regular investors begin to notice that the arbitrageurs are bringing down the value of all stocks, so they begin to sell too. Selling begets selling begets more selling.

According to the chair of the NYSE[31], computerized trading seems to be a stabilizing influence only when markets are relatively quiet. When the market is unsettled, programmed trading amplifies and accelerates the changes already underway, perhaps as much as 20%. Today the problem is arbitrage but, in the future, it is possible that ordinary investors will destabilize the market. This could conceivably happen because most investors will use the same type of computerized stock trading programs driven by very similar algorithms that predict nearly identical buy/sell points.

The question is, could these destabilizing effects occur in a world without computers? Arbitrage, after all, relies only on elementary mathematics. All the necessary calculations could be done on a scratch pad by any one of us. The problem is that, by the time we finished doing the necessary arithmetic for the stocks in our investment portfolio, the price of futures and the price of stocks would have changed. The opportunity that had existed would be gone.

EXAMPLE 5: Uniquely Cheap

Because computers can perform millions of computations each second, the cost of an individual calculation approaches zero. This unique property of computers leads to interesting consequences in ethics.

Let us imagine I am riding a subway train in New York City, returning home very late after a long day at the office. Since it is well past my dinner time, it does not take long for me to notice that everyone seated in my car, except me, has a fresh loaf of salami. To me, the train smells like the inside of a fine New York deli, never letting me forget how hungry I am. Finally I decide I must end this prolonged aromatic torture, so I ask everyone in the car to give me a slice of their own salami loaves. If everyone contributes, I can assemble a loaf of my own. No one can see any point in cooperating, so I offer to cut a very *thin* slice from each loaf. I can see that this is still not appealing to my skeptical fellow riders, so I offer to take an *arbitrarily thin slice*, thin enough to fall below anyone's threshold of concern. "You tell me how small it has to be not to matter," I say to them. "I will take that much and not a particle more." Of course, I may only get slices that are tissue-paper thin. No problem. Because I am collecting several dozen of these very thin slices, I will still have the makings of a delicious New York deli sandwich. By extension, if everyone in Manhattan had a loaf of salami, I would not have to ask for an entire slice. It would be sufficient for all the salami lovers to "donate" a tiny speck of their salami loaves. It would not matter to them that they have lost such a tiny speck of meat. I, on the other hand, would have collected many millions of specks, which means I would have plenty of food on the table.

This crazy scheme would never work for collecting salami. It would cost too much and it would take too long to transport millions of specks of salami to some central location. But a similar tactic might work if my job happens to involve the programming of computerized banking systems. I could slice some infinitesimal amount from every account, some amount so small that it falls beneath the account owner's threshold of concern. If I steal only half a cent each month from each of 100,000 bank accounts, I stand to pocket $6000 over a year's time. This kind of opportunity must have some appeal to an intelligent criminal mind, but very few cases have been reported. In one of these reported cases, a bank employee used a salami technique to steal $70,000 from customers of a branch bank in Ontario, Canada.[32] Procedurally speaking, it might be difficult to arraign someone on several million counts of petit theft. According to Donn Parker, "Salami techniques are usually not fully discoverable within obtainable expenditures for investigation. Victims have usually lost so little individually that they are unwilling to expend much effort to solve the case."[33] Even so, salami-slicing was immortalized in John Foster's country song, "The Ballad of Silicon Slim":

> In the dead of night he'd access each depositor's account
> And from each of them he'd siphon off the teeniest amount.
> And since no one ever noticed that there'd even been a crime
> He stole forty million dollars -- a penny at a time!

Legendary or not, there are at least three factors that make this type of scheme unusual. *First*, individual computer computations are now so cheap that the cost of moving a half-cent from one account to another is vastly less than half a cent. For all practical purposes, the calculation is free. So there can be tangible profit in moving amounts that are vanishingly small ***if*** the volume of such transactions is sufficiently high. *Second*, once the plan has been implemented, it requires no further attention. It is fully automatic. Money in the bank. *Finally*, from a practical standpoint, no one is ever deprived of anything in which they have a significant interest. In short, we seem to have invented a kind of stealing that requires no taking -- or at least no taking of anything that would be of significant value or concern. It is theft by diminishing return.

Does this scheme have a non-computer analog? A distributor of heating oil could short all his customers one cup of oil on each delivery. By springtime, the distributor may have accumulated a few extra gallons of heating oil for his own use. But it may not be worth the trouble. He may not have enough customers. Or he may have to buy new metering devices sensitive enough to withhold exactly one cup from each customer. And he may have to bear the cost of cleaning, operating, calibrating and maintaining this sensitive new equipment. All of these factors will make the entire operation less profitable. On the other hand, if the distributor withholds amounts large enough to offset his expenses, he runs the risk that he will exceed the customer's threshold of concern.

EXAMPLE 6: Uniquely Cloned

Perhaps for the first time in history, computers give us the power to make an exact copy of some artifact. If I make a verified copy of a computer file, the copy can be proven to be bit for bit identical to the original. Common disk utilities like *diff* can easily make the necessary bitwise comparisons. It is true that there may be some low-level physical differences due to track placement, sector size, cluster size, word size, blocking factors, and so on. But at a *logical* level, the copy will be perfect. Reading either the original or its copy will result in the exact same sequence of bytes. For all practical purposes, the copy is indistinguishable from the original. In any situation where we had used the original, we can now substitute our perfect copy, or vice versa. We can make any number of verified copies of our copy, and the final result will be logically identical to the first original.

This makes it possible for someone to "steal" software without depriving the original owner in any way. The thief gets a copy that is perfectly usable. He would be no better off even if he had the original file. Meanwhile the owner has not been dispossessed of any property. Both files are equally functional, equally useful. There was no transfer of possession.

Sometimes we do not take adequate note of the special nature of this kind of crime. For example, the Assistant VP for Academic Computing at Brown University reportedly said that "software piracy is morally wrong -- indeed, it is ethically indistinguishable from shoplifting or theft."[34] This is mistaken. It is not like piracy. It is not like shoplifting or simple theft. It makes a moral difference whether or not people are deprived of property. Consider how different the situation would be if the process of copying a file automatically destroyed the original.

Electrostatic copying may seem to provide a non-computer analog, but Xerox™ copies are not perfect. Regardless of the quality of the optics, regardless of the resolution of the process, regardless of the purity of the toner, electrostatic copies are not identical to the originals. Fifth- and sixth-generation copies are easily distinguished from first- and second-generation copies. If we "steal" an image by making a photocopy, it will be useful for some purposes but we do not thereby acquire the full benefits afforded by the original.

EXAMPLE 7: Uniquely Discrete

In a stimulating paper "On the Cruelty of Really Teaching Computer Science,"[35]Edsger Dijkstra examines the implications of one central, controlling assumption: *that computers are radically novel in the history of the world*. Given this assumption, it follows that programming these unique machines will be radically different from other practical intellectual activities. This, Dijkstra believes, is because the assumption of *continuity* we make about the behavior of most materials and artifacts does not hold for computer systems. For most things, small changes lead to small effects, larger changes to proportionately larger effects. If I nudge the accelerator pedal a little closer to the floor, the vehicle moves a little faster. If I press the pedal hard to the floor, it moves a lot faster. As machines go, computers are very different.

> A program is, as a mechanism, totally different from all the familiar analogue devices we grew up with. Like all digitally encoded

> information, it has, unavoidably, the uncomfortable property that the smallest possible perturbations -- i.e., changes of a single bit -- can have the most drastic consequences.[36]

This essential and unique property of digital computers leads to a specific set of problems that gives rise to a unique ethical difficulty, at least for those who espouse a consequentialist view of ethics.

For an example of the kind of problem where small "perturbations" have drastic consequences, consider the Mariner 18 mission, where the absence of the single word *NOT* from one line of a large program caused an abort.[37] In a similar case, it was a missing hyphen in the guidance program for an Atlas-Agena rocket that made it necessary for controllers to destroy a Venus probe worth $18.5 million.[38] It was a single character omitted from a reconfiguration command that caused the Soviet Phobos 1 Mars probe to tumble helplessly in space.[39] I am not suggesting that rockets rarely failed before they were computerized. I assume the opposite is true, that in the past they were far more susceptible to certain classes of failure than they are today. This does not mean that the German V-2 rocket, for example, can provide a satisfactory non-computer (or pre-computer) moral analogy. The behavior of the V-2, being an analog device, was a continuous function of all its parameters. It failed the way analog devices typically fail -- localized failures for localized problems. Once rockets were controlled by computer software, however, they became vulnerable to additional failure modes that could be extremely generalized even for extremely localized problems.

"In the discrete world of computing," Dijkstra concludes, "there is no meaningful metric in which 'small' change and 'small' effects go hand in hand, and there never will be."[40] This *discontinuous* and *disproportionate* connection between cause and effect is unique to digital computers and creates a special difficulty for consequentialist theories. The decision procedure commonly followed by utilitarians (a type of consequentialist) requires them to predict alternative consequences for the alternative actions available to them in a particular situation. An act is good if it produces good consequences, or at least a net excess of good consequences over bad. The fundamental difficulty utilitarians face, if Dijkstra is right, is that *the normally predictable linkage between acts and their effects is severely skewed by the infusion of computing technology*. In short, we simply cannot tell what effects our actions will have on computers by analogy to the effects our actions have on other machines.

EXAMPLE 8: Uniquely Coded

Computers operate by constructing codes upon codes upon codes -- cylinders on top of tracks, tracks on top of sectors, sectors on top of records, records on top of fields, fields on top of characters, characters on top of bytes, and bytes on top of primitive binary digits. Computer "protocols" like TCP/IP are comprised of layer upon layer of obscure code conventions that tell computers how to interpret and process each binary digit passed to it. For digital computers, this is business as usual. In a very real sense, ***all*** data is multiply "encrypted" in the normal course of computer operations.

According to Charlie Hart, a reporter for the *Raleigh News and Observer,*[41]the resulting convolution of codes threatens to make American history as unreadable as the Rosetta Stone:

- Historic, scientific and business data is in danger of dissolving into a meaningless jumble of letters, numbers, and computer symbols. For example, two hundred reels of 17-year-old Public Health Service tapes had to be destroyed in 1989 because no one could determine what the names and numbers on them meant.
- Much information from the past thirty years is stranded on computer tape written by primitive or discarded systems. For example, the records of many World War II veterans are marooned on 1600 reels of obsolete microfilm images picturing even more obsolete Hollerith punch cards.

This growing problem is due to the degradable nature of certain media, the rapid rate of obsolescence for I/O devices, the continual evolution of media formats, and the failure of programmers to keep a permanent record of how they chose to package data. It is ironic that state-of-the-art computer technology, during the brief period when it is current, greatly *accelerates* the transmission of information. But when it becomes obsolete, it has an even stronger reverse effect. Not every record deserves to be saved but, on the balance, it seems likely that computers will impede the normal generational flow of significant information and culture. Computer users obviously do not *conspire* to put history out of reach of their children but, given the unique way computers layer and store codes, the result could be much the same. Data archeologists will manage to salvage bits and pieces of our encoded records, but much will be permanently lost.

This raises a moral issue as old as civilization itself. It is arguably wrong to harm future generations of humanity by depriving them of information they will need and value. It stunts commercial and scientific progress, prevents people from learning the truth about their origins, and it may force nations to repeat bitter lessons from the past. Granted, there is nothing unique about this issue. Over the long sweep of civilized history, entire cultures have been annihilated, great libraries have been plundered and destroyed, books have been banned and burned, languages have withered and died, ink has bleached in the sun, and rolls of papyrus have decayed into fragile, cryptic memoirs of faraway times.

But has there ever in the history of the world been a *machine* that could bury culture the way computers can? Just about any modern media recording device has the potential to swallow culture, but the process is not automatic and information is not hidden below convoluted layers of obscure code. Computers, on the other hand, because of the unique way they store and process information, are far more likely to bury culture. The increased risk associated with the reliance on computers for archival data storage transforms the moral issues surrounding the preservation and transmission of culture. The question is not, Will some culturally important information be lost? When digital media become the *primary* repositories for information, the question becomes, Will *any* stored records be readable in the future? Without computers, the issue would not arise in this highly altered form.

So, this kind of example ultimately contributes to a “weaker” but still sufficient rationale for computer ethics, as explained earlier. Is it possible to take a “stronger” position with this example? We shall see. As encryption technology continues to improve, there is a remote chance that computer scientists may develop an encryption algorithm so effective that the Sun will burn out before any machine could succeed in breaking the code. Such a technology could bury historical records for the rest of history. While we wait for this ideal technology to be invented, we can use the 128-bit International Data Encryption Standard (IDEA) already available. To break an IDEA-encoded message, we will need a chip that can test a billion keys per second, throw these at the problem, and then repeat this cycle for the next 10,000,000,000,000 years.[42] An array of 10^{24} chips could do it in a single day, but does the universe contain enough silicon to build them?

CONCLUSION

I have tried to show that there are issues and problems that are unique to computer ethics. For all of these issues, there was an essential involvement of computing technology. Except for this technology, these issues would not have arisen *or* would not have arisen in their highly altered form. The failure to find satisfactory *non*-computer analogies testifies to the uniqueness of these issues. The lack of an adequate analogy, in turn, has interesting moral consequences. Normally, when we confront unfamiliar ethical problems, we use analogies to build conceptual bridges to similar situations we have encountered in the past. Then we try to transfer moral intuitions across the bridge, from the analog case to our current situation. Lack of an effective analogy forces us to discover new moral values, formulate new moral principles, develop new policies, and find new ways to think about the issues presented to us. For all of these reasons, the kind of issues I have been illustrating deserves to be addressed separately from others that might at first appear similar. At the very least, they have been so transformed by computing technology that their altered form demands special attention.

I conclude with a lovely little puzzle suggested by Donald Gotterbarn.[43] There are clearly many devices that have had a significant impact on society over the centuries. The invention of the printing press was a pivotal event in the history of the transmission of culture, but there is no such thing as Printing-press Ethics. The locomotive revolutionized the transportation industry, but there is no such thing as Locomotive Ethics. The telephone forever changed the way we communicate with other human beings, but there is no such thing as Telephone Ethics. The tractor transformed the face of agriculture around the world, but there is no such thing as Tractor Ethics. The automobile has made it possible for us to work at great distances from our local neighborhoods, but there is no such thing as Commuter Ethics.

Why, therefore, should there be any such thing as Computer Ethics?

---

[1]Parker, D. *Ethical Conflicts in Computer Science and Technology*. SRI International, Menlo Park, California, 1978.
[2]There was a follow-up study some years later that remedied some of the problems discovered in the original methodology. See Parker, D., Swope, S., and Baker, B., *Ethical Conflicts in Information and Computer Science, Technology, and Business*. QED Information Sciences, Inc., Wellesley, Massachusetts, 1990.
[3]Parker, D. *Crime By Computer*. Charles Scribner's Sons, 1976.

---

[4]Gotterbarn, D. The use and abuse of computer ethics. In *Teaching Computer Ethics*, Bynum, T., Maner, W., and Fodor, J., Eds. Research Center on Computing and Society, New Haven, Connecticut, 1991, p. 74.
[5]I coined the term "computer ethics" in 1976 to describe a specific set of moral problems either created, aggravated or transformed by the introduction of computer technology. By the fall of 1977, I was ready to create a curriculum for computer ethics and, shortly thereafter, began to teach one of the first university courses entirely devoted to applied computer ethics. By 1978, I had become a willing promoter of computer ethics at various national conferences. Two years later, Terrell Bynum helped me publish a curriculum development kit we called the "Starter Kit in Computer Ethics." We found we could not interest the academic establishment in computer ethics, either philosophers or computer scientists, but we managed to survive as an underground movement within the American Association of Philosophy Teachers.
[6]Weiner, N. Some moral and technical consequences of automation. *Science 131* (1960), pp. 1355-1358.
[7]Gotterbarn, D. A "capstone" course in computer ethics. In *Teaching Computer Ethics*, Bynum, T., Maner, W., and Fodor, J., Eds. Research Center on Computing and Society, New Haven, Connecticut, 1991, p. 42.
[8]Bynum, T. Computer ethics in the computer science curriculum. In *Teaching Computer Ethics*, Bynum, T., Maner, W., and Fodor, J., Eds. Research Center on Computing and Society, New Haven, Connecticut, 1991, p. 24.
[9]Moor, J. What is computer ethics? In *Metaphilosophy 16,* 4 (1985), p. 266. The article also appears in *Teaching Computer Ethics*, Bynum, T., Maner, W., and Fodor, J., Eds. Research Center on Computing and Society, New Haven, Connecticut, 1991.
[10]Maner, W. *Starter Kit in Computer Ethics*. Helvetica Press and the National Information and Resource Center for the Teaching of Philosophy, 1980.
[11]Pecorino, P. and Maner, W. The philosopher as teacher: A proposal for a course on computer ethics. In *Metaphilosophy 16,* 4 (1985), pp. 327-337.
[12]Parker, D. *Computer Crime: Criminal Justice Resource Manual*, 2nd edition. National Institute of Justice, Washington, D.C., 1989.
[13]Neumann, P. *Computer Related Risks*. Addison-Wesley Publishing Company, New York, 1995.
[14]Forester, T., and Morrison, P. *Computer Ethics: Cautionary Tales and Ethical Dilemmas in Computing*. MIT Press, Boston, Massachusetts, 1990.
[15]Tabak, L. Giving engineers a positive view of social responsibility. *SIGCSE Bulletin 20*, 4 (1988), pp. 29-37.
[16]Behar, J. Computer ethics: moral philosophy or professional propaganda? In *Technology in People Services: Research, Theory and Applications,* Leiderman, M., Guzetta., C., Struminger, L., and Monnickendam, M., Eds. The Haworth Press, New York, 1993, pp. 441-453.
[17]Chapman, G., in response to a luncheon address by Perrole, J., Political and social dimensions of computer ethics. Conference on Computers and the Quality of Life, George Washington University, Washington, D.C., September 14, 1990.
[18]See Gotterbarn, D. Computer ethics: responsibility regained. In *National Forum: the Phi Kappa Phi Journal 71,* 3 (1991), pp. 26-31.
[19]I refer to the academic discipline of computer ethics as defined in Maner (1980).

[20]Leventhal, L., Instone, K., and Chilson, D. Another view of computer science: patterns of responses among computer scientists. In *Journal of Systems Software* (January, 1992).
[21]Integrating the ethical and social context of computing into the computer science curriculum: an interim report from the content subcommittee of the ImpactCS steering committee. In *Proceedings of Ethicomp95: An International Conference on the Ethical Issues of Using Information Technology 2*, Rogerson, S. and Bynum, T., Eds. De Montfort University Press, 1995, pp. 1-19. For further information on the ImpactCS project, contact Dr. Chuck Huff in the Psychology Department at St. Olaf College in Northfield, Minnesota 55057 USA (huff@stolaf.edu).
[22]Neumann (1995), p. 88.
[23]Neumann (1995), p. 169.
[24]Moor (1985), p. 269.
[25]Green, P., and Brightman, A. *Independence Day: Designing Computer Solutions for Individuals with Disability*. DLM Press, Allen, Texas, 1990.
[26]See a similar discussion in Huff, C. and Finholt, T. *Social Issues in Computing: Putting Computing in Its Place*. McGraw-Hill, Inc., New York, 1994, p.184.
[27]Moor (1985), pp. 274-275.
[28]Neumann (1995), pp. 80-81.
[29]McConnell, S. *Code Complete: A Practical Handbook of Software Construction*. Microsoft Press, Redmond, Washington, 1993.
[30]Science behind the news: Did computers make stock prices plummet? In *Discover 7*, 12 (December, 1986), p. 13.
[31]Computers amplify black Monday. In *Science 238*, 4827 (October 30, 1987).
[32]Kirk Makin., in an article written for the *Globe and Mail* appearing on November 3, 1987, reported that Sergeant Ted Green of the Ontario Provincial Police knew of such a case.
[33]Parker (1989), p. 19.
[34]Quoted in Ladd, J. Ethical issues in information technology. Presented at a conference of the Society for Social Studies of Science, November 15-18, 1989, in Irvine, California.
[35]Dijkstra, E. On the cruelty of really teaching computer science. In *Communications of the ACM 32*, 12 (December, 1989), pp. 1398-1404.
[36]Dijkstra (1989), p. 1400.
[37]Neumann, P. Risks to the public in computers and related systems. *Software Engineering Notes 5*, 2 (April, 1980), p. 5.
[38]Neumann (1995), p. 26.
[39]Neumann (1995), p. 29.
[40]Dijkstra (1989), p. 1400.
[41]Hart, C. Computer data putting history out of reach. *Raleigh News and Observer* (January 2, 1990).
[42]Schneier, B. The IDEA encryption algorithm. *Dr. Dobb's Journal*, 208 (December, 1993), p. 54.
[43]Gotterbarn (1991), p. 27.