Walter Maner

Is Computer Ethics Unique?

[An earlier version of this paper appeared in *Science and Engineering Ethics*, volume 2, number 2 (April, 1996), pages 137-154. See http://www.cableol.co.uk/opragen/ Used by permission.]

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. This may be true, but the perceived need for remedial moral education does not 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 it can provide a 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, 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 and 1985, 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, Neumann, Forester and Morrison 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.

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

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 also worry about the impenetrable wall of computer-generated privacy afforded to every computer-literate criminal.

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 twenty years, we have not been able to assemble a critical mass of self-defining core issues.

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.

Does this case have a satisfactory non-computer analog? Consider an automobile’s mechanical odometer gauge. When the odometer reading exceeds a designed-in limit, say 99,999.9 kilometers, the gauge overflows and returns to all zeros. This is a non-computer analogy, but not a satisfactory one. Perhaps it would be satisfactory 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 incident at the hospital in Washington, D.C. meets my three basic requirements for a unique issue or problem. It is characterized by the primary and essential involvement of computer technology, it depends on some unique property of that technology, and it would not have arisen without the essential involvement of computing technology.

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."

The unique adaptability and versatility of computers have important moral implications. Since computers do not care how they get their input, there is nothing to prevent a paraplegic from using a head-controlled pointing device to send information to a computer. On Kantian grounds, we have a clear duty to modify standard input devices to solve accessibility problems, 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 a paraplegic 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.

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. 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.

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*. 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.

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, 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.

Suppose my job is to program the computers that manage bank accounts. I could write the program so that it moves a tiny amount from every account into an account I own. I could make the amount so small that it would fall beneath the account owner's threshold of concern. Even 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. There are at least three factors that make this "bit shaving" or "salami slicing" 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. *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.

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." 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,"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.

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. 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. It was a single character omitted from a reconfiguration command that caused the Soviet Phobos 1 Mars probe to tumble helplessly in space. 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." 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.

Because of this common practice, much valuable information from the recent past is stranded on computer media that can only be deciphered by primitive or discarded systems using obsolete software. This growing problem is due to 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.

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.