This is a slightly revised version of an article published in *Software Development*, March 2000. For the published version see

www.sdmagazine.com/documents/s=746/sdm0003d/0003d.htm?temp=N3TyuRoNcQ

(Registration required, but possible for free.)

The Ethics of Free Software

Bertrand Meyer, March 2000

The movement in favor of free and open-source software has recently reached a highly visible status, not only in the computer profession but in the popular media, with mass-circulation magazines as widely available as Time and Newsweek giving prominent coverage to such heroes of the movement as Richard Stallman, Eric Raymond and Linus Torvalds. Comments on free software in its various form have overwhelmingly been laudatory, hailing the approach for its selflessness, little attention has been devoted to justifying this assessment and, more generally, to explore the associated ethical issues.

In this article I will try to correct this situation by taking a close look at the moral aspects of free and open-source software.

1. ABOUT ETHICS

It is useful first to define a basis for this discussion by recalling some principles of ethics. I will stay away from controversial aspects, limiting the list to generally recognized ideas.

- Ethics is about right and wrong. Human beings are equipped with an ability, partly innate and partly acquired, to judge human actions as morally good or bad. This does not mean that “right” and “wrong” mean exactly the same for everyone, simply that everyone possesses a notion of right and wrong.
- In spite of the diversity of moral views, ethics includes a universal component. The differences are not only individual (my next-door neighbor may find repellent a behavior which to me is perfectly acceptable) but also cultural: for example in Western countries we find it normal that a woman should show her face in public, a conduct that in some parts of the world is considered immoral. But many principles are culture-independent. Killing an innocent person, for example, is not morally acceptable, regardless of your culture. This discussion will assume that there are a number of such moral absolutes and will ignore more specific principles.

- Ethical rules may be rated as more or less important. Most people would accept all of the following as consequences of moral imperatives: do not litter a public park by disposing of a paper napkin on the lawn; if you are sitting in a bus and see that an obviously tired elderly person cannot find a seat, cede yours; do not steal from the corner grocery store; do not kill the next person that you see walking peacefully down the street; do not send tanks, troops and aviation to grab land from your neighboring country. But some of these are more fundamental than others, meaning in particular that corresponding violations are more repulsive than others (perhaps, for these examples, in the order given).
- Human beings have both good and bad impulses. A society in which everyone is evil all of the time would be impossible to manage; but in no society is everyone good all the time.
- One should judge people on their actions. Intentions can sometimes provide extenuating circumstances, but what counts is what people voluntarily do, not the reason they have, or invoke, for doing it. Hell is paved with good intentions. Aside from other reasons, limiting ourselves to judging deeds, not thoughts, is easy to justify on purely pragmatic grounds: you can observe my actions, or at least their results; you cannot tell whether my excuses are real or imagined.
- Not causing unjustified loss of human life is one of the universal moral imperatives.
- Not damaging someone else's reputation through misrepresentation is another.
- Not acquiring someone else's legitimate property against his will is yet another.
- Justly remunerating someone else's services is related to the preceding one.

The word "justly" makes the last principle perilous to apply in practice, since fairness is a subjective concept. Since this discussion is only about ethics, we cannot resolve the issue fully because this would involve economics (in a market economy, fairness is a tradeoff between demand and supply). You, my publisher, may offer me $100 for my latest novel, adding that it's really boring and you are doing me a favor, whereas I, having spent five years on it and being convinced that the result is —frankly — brilliant, scream that anything less than a million would be a moral outrage. Ethics won't bring a resolution here, only economics and the strength of our respective negotiating positions. But if I accept your proposal and the next day you sell the rights to Steven Spielberg for $10 million, many people will consider that you have done me wrong ethically.

An area perhaps more directly related to ethics than economics is the law. In an ideal world, there might be perfect identity between the legal and the moral. But not in the real world. What's legal is not necessarily moral: it is not very moral to say nasty things to your mother on her birthday, but you're not violating any laws. What's moral is not necessarily legal: not so long ago Southern US states forbade a white person from bringing a black acquaintance into a "whites only" public place, contradicting the obvious moral imperative that one should not discriminate against people because of the color of their skin. It would be exaggerated and cynical to infer that there is altogether no connection between ethics and the law; but at most what the law will give us (even if we exclude obviously immoral laws like the last one cited) is, rather than an absolute criterion, a rough indication, valid for a particular place at a particular time, of practices deemed socially desirable, partly for moral reasons and partly for others.

2. ABOUT FREE SOFTWARE

Even among the proponents of Free and Open Source software the definition of these terms is a constant source of controversy. For example the GNU project (at http://www.gnu.org/philosophy/free-software-for-freedom.html) chides Eric Raymond's Open Source initiatives for attempting to trademark the term "open source". It also criticizes many providers of free software such as Apple (see www.gnu.org/philosophy/apsl.html), the Berkeley Unix Software distribution (bsd.html in the same directory) and Netscape (netscape.html) for not observing the exact GNU definition of "free", or using license terms different from those of GNU. It would thus be futile to attempt to provide a comprehensive definition here. For the present discussion, however, the general scope is clear; we consider software that:

- Is legally available from without payment from at least one source (which does not preclude other sources from offering it for payment, for example to people who want a distribution on CD rather than downloaded, or require commercial support).
- Can be used for commercial as well as not-for-profit development, even by people who have not paid for it. (There may be some restrictions on commercial uses, for example the requirement that additions to the free software be free too.)
- Can be obtained in source code form, and hence modified.

Until it arrives at a more precise set of definitions (in section 3), this discussion will use the term "free software" to denote products that satisfy the properties listed above. Note that this is not the definition promoted by the GNU project, which, as we will see, uses "free" in a much stronger sense. The intent is not to fight over terminology, but simply to make do with the limited number of terms in the English language. In addition, the rapidly growing interest in "free" software extends to many products that would not be considered free under the GNU definition, but fall well within the one above.

The growth of free software has indeed been remarkable in the past few years. One of the most visible results has been the Linux operating system, developed under the leadership of Linus Torvalds and nominally available at no cost (although usually installed from a CD obtained, for a price, from a commercial company). Other widely used free tools — some having for-a-fee variants too — include the TEX and LATEX document processing utilities, a quasi-standard for preparing theses in computer science departments; GCC, the GNU project's C and C++ compiler; Apache, one of the most widely used Web servers; the FreeBSD operating system kernel, a Linux competitor; the EMACS and VIM text editors. Through these and other products, the free software movement has already established an indelible mark on the computer industry.

Perhaps the most striking aspect of this movement is the worldwide availability of often talented software developers willing to contribute their time, energy and creativity to free software.

It is this pool of enthusiastic contributors, willing to work even without immediate monetary reward, that has led in part to the general feeling of goodwill surrounding the free software community.

It should be pointed out, however, that the existence of a community of dedicated, well-intentioned and sincere defenders of a cause is unrelated to the ethical value of that cause. As an example, one of the tragedies of the twentieth centuries has been the diversion of the energy and passion of countless honest and idealistic volunteers towards support for Soviet-style communism, a regime that cause tens of millions of deaths, uncounted cases of human misery, and the destruction of civil society in entire countries. This example is obviously not a comparison with the free software community, simply a reminder that no idea can be justified on the basis of the quality of its supporters. The observation works the other way too: bad people can defend good causes. A corrupt and dishonest politician may sincerely support principles of democracy and freedom. His personal failings do not disqualify the ideas of democracy and freedom any more than the Nazi regime's impressive building of autobahnen disqualifies the merits of freeways.

3. THE ECONOMICS OF FREE SOFTWARE

The term "free software" (with the earlier definition, implying availability at no cost) is almost always a misnomer since software is produced by humans and in modern societies no human can live without money; so even if no one paid for the software someone must have paid the software developer. In practice the only possible cases are the following:

- The software developer may have a personal fortune freeing him from the monetary concerns of most of his fellow human beings. Although this case may have occurred in connection with free software, I am not aware of any example. We may treat it (if it arises) as a special case of the next one.
- The software developer may have other sources of income, paid by an employer or client as compensation for services unrelated to the free software. The developer may then, in his own time and using his own resources, without direct or indirect participation of the client or employer, and without violating agreed restrictions on external activities, develop products that he makes available as free software. As will be clear from the other cases, this is in the only case in which it is really appropriate to talk of "free software", although this term is in fact too weak; "*donated* software" would be more accurate.
- Many public institutions such as universities will release for general use most of the software developed by their employees (although, as universities around the world are being pressed by the purse-string holders to enhance their economic value, and recognize the economic potential of the software they develop, this generous attitude is not as universal as it used to be). In this case the software may be free to its users but it was not free to produce. "Free software" is a misnomer; one should talk, in the case of public universities, of Taxpayer-funded software. For example the GNU Eiffel compiler was developed at the University of Nancy by employees of that university who (in contrast with commercial Eiffel vendors, who need paying customers to survive) get every month a salary from the state, whether the users are happy or not with the product. This is a typical case of taxpayer-funded software.
- Companies may find it beneficial to release some of their software products without asking for a fee. The goal may be: favorable publicity; an attempt to establish the company's chosen solution as the standard, thus gaining a competitive edge and moving technologically one step ahead of the competition; a desire to get rid of maintenance costs and efforts by offloading the work to unpaid (or more commonly paid by someone else) volunteers; a sincere effort to help the community; several of the above. In this case we should again not talk of free software; the phrase should be "*privately funded* software".
- It may be useful to define a special case of the previous two for situations in which the developers were not officially required by their supervisors to develop the software, but did it anyway on company time, or using company resources, or both, and were then authorized to release for free. For example Richard Stallman's description (at

> www.april.org/actions/rms/10111998/texte.html) of the origins of his EMACS editor, one of the most visible early examples of free software, makes it clear that this happened as part of his job as systems programmer at the Massachusetts Institute of Technology. Although this is very close to the previous two cases, we may use the more specific terms "Taxpayer-sponsored software" and "Privately-sponsored software". Even the icon of free software, GNU, started in the same way according to Stallman (on the page just cited)1:
>
>> *"So I resigned from MIT and I started to write the GNU system in January of 1984. Fortunately the lab's director allowed me to continue using the lab's computers. I assume that I could have found machines elsewhere, since the EMACS program I had written was rather respected, but that way I didn't have to look."*
>
> The story goes on to state that Stallman later "resigned" — presumably meaning that he stopped using the MIT's machines, since it appears from the above that he had already resigned — because "sometimes, universities take software written by their employees to sell them as proprietary products". (What a shame indeed: that a university would think it has any rights at all on products developed by people it pays, on machines that it owns!)

The categories identified here — donated, taxpayer-funded, privately-funded, taxpayer-sponsored and privately-sponsored — seem to exhaust the economic possibilities; they provide precise and accurate terminology, more useful in practice than the catch-all term of free software.

4. VOICES FROM THE REVOLUTION

Many of the contributions of the free software community are admirable. Highly disturbing, however, is its common hatred and slander of the commercial software world.

Richard Stallman from GNU and the Free Software Foundation (FSF), the best-known figure of free software, professes an absolute refusal of any notion of commercial software. Software should be free, period. A few samples from the GNU and FSF web pages include:

- "Signing a typical software license agreement means betraying your neighbor: 'I promise to deprive my neighbor of this program so that I can have a copy for myself.'" (www.gnu.org/philosophy/shouldbefree.html.)
- "When a program has an owner, the users lose freedom to control part of their own lives." (www.gnu.org/philosophy/why-free.html.)
- "The system of copyright gives software programs 'owners', most of whom aim to withhold software's potential benefit from the rest of the public. They would like to be the only ones who can copy and modify the software that we use." (Same URL.)
- "I think that to try to own knowledge, to try to control whether people are allowed to use it, or to try to stop other people from sharing it, is sabotage. It is an activity that benefits the person that does it at the cost of impoverishing all of society. One person gains one dollar by destroying two dollars' worth of wealth. I think a person with a

[1] The reference cited is a transcript of a lecture given in Paris by Dr. Stallman, in French, in November of 1998. The citation, as well as two later ones from the same source, is a literal translation into English.

> conscience wouldn't do that sort of thing except perhaps if he would otherwise die." (BYTE interview, www.gnu.org/gnu/byte-interview.html.)

And so on (there are countless other examples). These are strong indictments, based on moral terms. They are morally unjustifiable. Nowhere in the hundreds of pages of GNU and FSF literature is there any serious explanation of why it is legitimate, for example, to make a living selling cauliflowers, or lectures (as a professor does), or videotapes of your lectures, but criminal to peddle software that you have produced by working long hours, sweating your heart out, thinking brilliantly, and risking your livelihood and that of your family.

This absence of rational justification for the extremist view that all commercial software is evil is all the more striking given that some other parts of the GNU/FSF literature can be serious and reasoned. Its criticism of software patents, for example, is often cogent, and takes the trouble of presenting the opposite view to refute it. As soon as the discussion is about free software — and that's where it is much of the time — argument yields to excommunication.

The only stated justification for the indictment of commercial software — apart from nostalgic reminiscences of how nice life was in the early days of the MIT Artificial Intelligence Laboratory, and how horrible it became when printer manufacturers started distributing the software in binary form, a tale that may elicit sympathy from the reader but hardly has any universal moral value (I too remember fondly when you could get access to US National Parks for free by just showing a foreign passport, but that doesn't mean the National Park Service has suddenly turned evil) — is that software is different from other wares since it can be reproduced so easily. But this does not stand a minute's scrutiny. The difference is a matter of degree, not nature; software reproduction always costs something, even if it is as little as a dollar for a CD, one cent of network connection time for an Internet download, or the marginal cost of using up more memory. With a good scanner or photocopier, you can reproduce a book, too, for very little money these days. Yet another example is TV signals from satellites, which unlike software are in fact truly free to reproduce once you have paid for your own antenna and receiver. We may grumble at having to pay for a mere wave in the ether, but is it immoral? Most people don't think so, accepting instead that it would be immoral to obtain the contents of the signals without economic compensation to the people — producers, actors, technicians — who worked to produce it.

In any case the idea that a low reproduction cost should imply a free product has no rational basis. In fact no known moral law implies that purchase cost should even be related to production cost. I may find ridiculous the idea of paying eight times as much for a BMW as for a Toyota Corolla if I guess that it costs far less than eight times as much to produce; but that doesn't make BMW guilty of moral horrors. This is a purely economic issue (how much is prestige worth to the buyer?).

A social philosopher preoccupied with fairness might argue that, as every American is entitled by birth to an affordable car, denying it is a moral outrage; but this is unrelated to the issue since Toyota's cheaper prices are presumably due not to superior social awareness but to a self-serving business strategy — the decision to target a different market.

The GNU and FSF view is that it is OK to sell anything except software. (To be precise, I have not found any example of something else whose selling they find immoral; satellite signals might seem a logical candidate.) Computers are OK; services are OK. But if the work of your life is a great software package, trying to make a living out of selling it — unless you also give it away, an immediate business-killer — is a moral abomination. It should be pointed out here that history suggests the reverse moral lesson. Until the 18th century, writers were ripped off by publishers. The gradual imposition of a copyright (due largely in France to Beaumarchais, author of the Barber of Seville and the Marriage of Figaro as well as smuggler of arms to the

American Revolution) was a major moral correction, re-establishing the rights of the creators. The new idea was that the "software" (the abstract contents) had a value, not just the "hardware" (the actual paper, leather and ink making up a physical book). One of the first authors to benefit from this new economic order was Voltaire, who used his great wealth, arising originally from the sale of his books — his software —, to fund his lifelong fight against tyranny and injustice. The extremist free-software view would have us return, for software, to a pre-eighteenth-century world: you can make money from selling CDs, but cannot protect the contents of those CDs!

GNU's and FSF's use of moral terms to indict commercial software providers, without any moral basis, relies on a perverse distortion of language. The authors take pains to explain that "free" means not cost-free but available for modification by anyone. Free as in free speech, the cliché goes, rather than free beer. This is acceptable: after all, the word "free" is ambiguous in English, whereas other languages distinguish between liberty (as in *libre*) and zero cost (as in *gratuit*). It is common, in scientific and technical discourse, to use a term from everyday language in a specialized sense, as long as you state your definition precisely and stick to it. But the use of "free" and "freedom" in the GNU literature is far from neutral. Much of that literature is in effect a pamphlet demanding "freedom" for software developers. The GNU license itself reads not like a license but like a manifesto against the evils of proprietary software. These passionate speeches, based on a specialized notion of "freedom", use the universal appeal of this word, derived from centuries of humankind's struggle for freedom in the usual (political and moral) sense of the term, to defend the authors' own agenda, based on a narrow and controversial notion of freedom.

This distortion — the hijacking for private purposes of a word that holds such a sacred aura for most people — is unethical.

Extreme analogies are another dubious rhetorical device. One of the first comments one encounters in the GNU pages (www.gnu.org) is a comparison to the Soviet Union:

> *"All four practices* [of the Software Publishers Association, to prevent theft of software] *resemble those used in the former Soviet Union, where every copying machine had a guard to prevent forbidden copying, and where individuals had to copy information secretly and pass it from hand to hand as 'samizdat'."*

There is a comic side to such pronouncements; hearing the epitomes of capitalism — Microsoft and such — accused of bolshevism by a group that advocates collective property of all software seems bit far-fetched. In fact Bob Metcalfe, in a recent *InfoWorld* column, did not hesitate to write that "Richard Stallman is a communist". I do not actually think such comments are particularly useful; the best way to counter the sometimes outrageous attacks of the most extreme "freedom" advocates may be to keep a cool- headed, rational attitude, and not try to match their antics. (Dr. Stallman himself said, in response to a *Byte* interviewer, that he is neither a socialist nor a communist.)

It would all the same be a mistake to portray that group as slightly eccentric do-gooders. Their propaganda is a campaign of hatred against people whose only "crime" is to want to make a living out of the wares they produce. A recent encounter with Richard Stallman illustrates this attitude. Mutual friends had thought it a good idea to bring him and a commercial software developer to a dinner party in a restaurant. The software developer, curious about the idea of free software and interested in sharing ideas with a creative colleague with a different background, tried to open a friendly discussion with: "*I am a commercial software developer, but I appreciate much of your work and have in fact recently changed the terms of our free software license as a result of your observations*". Unfortunately, he never in the entire evening got to the "*but*", as the opening triggered an explosion of abuse to the effect that in such a case there was not even anything to discuss, commercial software being the most horrible thing on earth and a denial of everyone's right to freedom. The rest of the evening, needless to say, was rather painful, as no one likes friendly overtures to be met with violent rebuke. It also led the commercial software developer, who until then had not been unreceptive to the quasi-universal eulogy of free software, to a much more carefully researched assessment of the pros and cons of the movement.

It is unfair, of course, to judge an idea from the character of its proponents. But in the case at hand the connection is close, as Dr. Stallman is the living icon of the free software movement, widely admired, imitated and idolized (almost like a sect leader) by his followers; he is also listed as the author of much of the GNU literature — the only one, in fact, in the documents that I have seen. So his attitude shapes much of the free software community's perception of commercial software.

That perception is that commercial software is evil. In fact, one of the catchy GNU links is entitled "*Is Microsoft really the great Satan?*". The page to which it leads (www.gnu.org/philosophy/microsoft.html) gives the answer. In short, yes indeed, but don't just pick on Microsoft; anyone else who "*denies users their rightful freedom*" (i.e. sells software) is just as satanic.

5. A SKEWED MORAL PERSPECTIVE

The preceding citations indicate that free software advocates condemn commercial software vendors (meaning most of the industry) on the basis of moral absolutes. As later citations will show, they present themselves as generous benefactors of humanity. What is striking is the reduction of all software-related moral issues to one aspect, a special notion of "freedom", as if nothing else mattered. This leads to pronouncements that would be funny if they weren't also scary at times. Dr. Stallman writes at www.april.org/actions/rms/10111998/texte.html:

> "[...] *scientists used sometimes to be able to cooperate even when their countries were at war. I read that once American soldiers who landed on an island of the Pacific Ocean, during World War II, found a building with a note saying: 'To American soldiers. This building is a marine biology laboratory. We put all our samples and reports in order so that US scientists can go on with our work.' Because, for them, they were working only for humanity. Not only for Japan. They wanted their work to be useful for humanity, regardless of the outcome of the war. But today we live in a state of civil war between small groups, in every country. In which every group acts to stop the others, hinder the others, hamper the others. It's sad."*

Ethically, this is appalling. In the absence of a reference it is not clear where the anecdote comes from; but to cite as an ideal, without so much as a qualification, one alleged act of scientific chivalry from a country whose troops conducted World War II as an series of atrocities including countless war crimes, and to claim that the situation of today's peaceful industrialized societies is "*sadly*" worse (a "*civil war*") because some selfish people refuse to share the source of their software, shows a noxious contempt for the hierarchy of ethical values. What would a survivor of the horrendous Singapore POW camps or the Rape of Nanking think of Dr. Stallman's little miseries resulting (as described a few lines down from the above extract) from computer manufacturers that

> *"took the X Window system [originally free software from MIT], compiled it for their system, and distributed copies as a proprietary product with exactly the same lack of freedom as with Unix."*

This is the danger of single-issue proponents: they lose all sense of perspective and start thinking that their perceived "moral" problem is the only one that counts.

Here is another example of the consequences. Eric Raymond, another of the leaders of the free software movement — although he prefers the term "open source" — uses his Web page to proselytize for gun rights. One quote will suffice, although readers interested in this propaganda can find it at www.netaxs.com/~esr/guns/gun-ethics.html and neighboring pages. The title is Ethics from the Barrel of a Gun: What Bearing Weapons Teaches About the Good Life; note the reference to ethics. It starts:

> *"There is nothing like having your finger on the trigger of a gun to reveal who you really are. Life or death in one twitch — ultimate decision, with the ultimate price for carelessness or bad choices. It is a kind of acid test, an initiation, to know that there is lethal force in your hand and all the complexities and ambiguities of moral choice have fined down to a single action: fire or not?"*

Such balderdash would be easy to dismiss if it were not highly visible from the author's Open Source pages (I came across it when looking for Mr. Raymond's famous essay, *The Cathedral and the Bazaar*) and didn't have any ethical implications.[2]

A bit of background is useful for non-US readers. A minority of gun nuts (a term that Mr. Raymond actually applies to himself), supported by an all-powerful lobby, the National Rifle Association, has managed to terrorize Congress into maintaining loose gun laws with no equivalent in the rest of the civilized world. The official pretext — the Constitution's Second Amendment, devised in the late 18th century to establish a popular, Swiss-style militia guarding against the return of oppressive power — is absurd for many reasons: there is no Swiss-style militia in the US (guns are used for pleasure and, of course, for killing ordinary people); the US political system has a remarkable combination of checks and balances, making the imposition of a dictatorship rather unlikely; the historical exceptions to this observation — such as McCarthyism and institutionalized racial discrimination — were not, if memory serves us well, met by armed resistance from an outraged citizenry; and a real aspiring dictator would have means of oppression, such as missiles, tanks and perhaps nuclear weapons, against which even the sophisticated guns on which Mr. Raymond roves ecstatic in his Web pages would be rather powerless.

But the results of such a stance are clear to everyone, and a shock to citizens of other democratic countries: a murder rate higher than in any other first-world country, a constant race between police and criminals for ever more lethal weapons, free availability of murderous devices in barely regulated "gun shows", 12-year-olds trained in weapons since kindergarten who go on shooting rampages with guns borrowed from the family cupboard. Month after month, the reports of death and devastation, most avoidable with the common-sense rules in place in other countries, hit the headlines. To which the "gun nuts" respond with: "*Guns don't kill people, people do*". (Sure. Try murdering fifteen of your co-students and five of your teachers in five minutes with a kitchen knife.)

Is it right, one might ask, to make a connection between Mr. Raymond, who is only one person, and the rest of the free software community? Although some readers will disagree I believe the answer is yes, for at least three reasons:

[2] This passage about Raymond's gun advocacy caused a lot of criticism from some readers, as the issue is highly controversial in the US. I have left it for completeness, but it is only a side argument; the reader who doesn't agree with me on this point can ignore it, as the rest of the discussion stands without it.

- His propaganda is prominent in his Web pages, one of the most frequently accessed sources of information about the free software movement, to which the media's references draw countless unsuspecting visitors.
- Eric Raymond has been one of the most visible proponents of the Open Source movement, widely interviewed and cited. He is a public person; his views inevitably commit the rest of the movement unless it disavows them.
- They have not indeed, as far as I know, been publicly disavowed by other open source leaders; for example Richard Stallman, whose differences and competition with Eric Raymond are widely known, has not to my knowledge — dissociated himself from the gun propaganda.

These other leaders would be well-inspired in my opinion to dissociate themselves from Raymond's view of freedom, and confirm that it's not what they have in mind. Given the choice between

- a society where all software would be proprietary, and civilized measures would be in place preventing (for example) a disturbed white supremacist from buying a police gun without any background check at a gunshow, then going to a Jewish day camp in Los Angeles to shoot at everyone in sight (a tragedy that happened just a few weeks ago);
- a society where all software would be free and Mr. Raymond's views on gun "freedom" were fully realized

any ethically conscious person would choose the former. The free software advocates should acknowledge that some issues are more important than who owns software, and that human life is one of them.

6. THE QUALITY CONCERN

One issue that is less important than human life (except that it may ultimately affect human lives) but has obvious ethical resonances is the quality of the software a programmer produces.

Isn't it relevant, for an ethics-preoccupied software engineer, to worry about this aspect? Is it more or less important than software ownership?

In a way, it is unfair to take the free software advocates to task on this issue because part of the success of offerings such as Linux and GCC has been their reputation for quality, and the often repeated comment that, if you find a bug in one of these products, you will have a much easier time reporting it and getting it fixed than if you try calling Sun[3] or Microsoft customer support about a problem with their proprietary, binary-only products. With open source you benefit — so the explanation goes — from the collective ability of many people to dive into the code and find out what's wrong.

This reasoning is largely correct, at least for some of the major free software products. Linux and GCC are widely praised by their users. Yet not all is rosy. Like commercial software, free software is — surprise — of very variable quality. You find the best and the worst. My own experience with free software has included both kinds. Recently, my company has had more than our share of the second; we have had to cancel one major project, and reengineer a product completely, after wasting many person-months and disappointing customers, because of the deficiencies of two separate GNU products (the GCC compiler for Windows and the editor

[3] Since the appearance of this article a significant part of Sun's software has been open-sourced.

under GTK). In both cases the scenario was the same: fixes to well-known bugs being promised and promised again; everyone waiting for months and months, until it becomes clear that nothing will happen; in the end, having to write off all the affected developments. Since no one is in charge, and you didn't pay for the products, there is no one to blame.

Even in products that are good or excellent overall, the distribution of work among many people may mean a high degree of variation between components of a product. One person who doesn't share the general enthusiasm for Linux is Ken Thompson from AT&T, co-creator of the original Unix and recipient of the Turing award (the "Nobel prize" of computing). In a recent interview he couldn't contain his scorn for the quality of the Linux code:

> "*I've looked at the source, and there are pieces that are good and pieces that are not ... My experience and some of my friends' experience is that Linux is quite unreliable. Microsoft is really unreliable but Linux is worse.*" (*IEEE Computer*, 32, 5, May 1999, page 61.)

In a different case, the newsgroups *comp.risks* recently published a report of rather horrendous and elementary C errors found in a quick and simple check of the source of the FreeBSD operating system (see catless.ncl.ac.uk/Risks/20.18.html#subj9.1).

Even though the GNU products are often good, the licenses which accompany them are no better, in the warranties (or rather absence thereof) they offer to the user, than commercial software. A typical quote, from www.gnu.org/software/year2000.html:

> "*The Free Software Foundation does not provide warranties for its software. We can't afford to. So we can't promise that GNU software has no Year 2000 bugs, any more than we could promise you the same thing about another sort of bug.*"

It is indeed amusing to note, in the GNU public license, the place where the inflammatory proclamations about freedom stop and the tone becomes much more measured. That is also — surprise! — the place where the license switches to the topic of software "warranties". At this point the discourse begins to sound much more familiar (upper case in the original, see www.gnu.org/copyleft/gpl.html):

> "*EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.*"

Doesn't this sound familiar? Where have we seen it before? Oh yes! In the hated "proprietary" licenses, those which "*take away your freedom and mine*": no warranty of any kind; "*AS IS*"; caveat emptor.

In using such terms the GNU license is neither better nor worse than most of the rest of the industry. Very few providers of software are able, today, to warrant their software the way producers of non-software artifacts warrant their offerings. (It should be pointed out, however, in reference to the first of the quotations above, that Y2K-compliance guarantees are becoming increasingly available.)

We cannot be too harsh on GNU and consorts in this respect. But given the violence of their attacks on the rest of the software industry, and their self-positioning as freedom-defending angels, it is fair to ask: do you have your priorities right?

Here is a hypothetical question to help understand what's at stake. Assume two pieces of software with a similar purpose. We could imagine for example that they are both for managing a surgical device or airplane controls, but let's not make things unnecessarily dramatic and simply assume a normal software application, important and useful but not necessarily life-threatening — a text editor, a compiler, a graphics library, a Web server. Now assume that the two products differ as follows:

- Product F is free software. It comes with the standard no-warranty warranty.
- Product P is proprietary software. It costs $50 for the binary-only version. It uses the most advanced techniques of software engineering. It never crashes, or departs in any way from its (mathematically expressed) specification. The seller is, in fact, so sure of those qualities that he will commit in writing that any violation of the specification during execution will immediately lead to reimbursement of the purchase price and compensation for any damages incurred.

According to the free software literature, product F, being the only one that preserves users' "freedom", is the ethical one. The seller of product P is still a repugnant profiteer.

The question, left as an exercise for the reader, is: which one of these solutions do you consider the more ethical? Auxiliary question: does your answer change if the price of the product becomes $5000? $50,000? $5 million?

Hypothetical though they may be (an unfortunate result of the current state of the art in software engineering), these are bona fide questions. I think I know my own answers: I would consider the second solution more ethical (certainly at $50, and possibly regardless of the price, unless perhaps someone can show that it takes advantage of an undue monopoly), but I don't claim these are the only possible answers. The free software advocates, for their part, do sound like they have found the truth. What I mostly see is that by concentrating on one aspect of the picture they have developed a skewed view of ethics, which can, by itself, become unethical.

Second auxiliary question for special credit: you think the revolutionary software development technique you have just invented makes it possible to produce provably bug- free products such as P. You need $20 million dollars to productize it. Your university will be glad to pay for a postdoc for six months if you teach an extra course. The government funding agencies, after taking a year to review your application, tell you to get lost as your proposal has no commercial value. (Public funding is very much "goal-oriented" these days.) But you have found a group of investors that values your idea. Of course they expect to make a huge windfall from their investment, so they'll laugh if you suggest free software.

The question: should you go with them, or is it more ethical to turn down the proposal and devote the next five years of your life to a free LATEX extension for Babylonian cuneiform?

7. PRODUCTS AND SERVICES

If we accept that the matter of software quality may have ethical implications, one of the tenets of the free software movement becomes rather disturbing. The free software axioms hold, as we have seen, that although charging for software is wrong it is all right to charge for services associated with the software, such as maintenance and training. The risk here is that such an attitude may lead to products with known deficiencies, giving the provider a ready-made source of juicy service contracts.

To date, there is no proof that any provider of free software has engaged in such practices, so it would be unfair to turn this worry into a criticism of the current state of the art in free software. But as a worry it is legitimate, especially at a time when ever bigger fish from the mainstream software industry clamor their newly discovered fondness and enthusiasm for the gospel of free software (proclamations which, in some cases, cannot but leave a strange aftertaste).

The practice of releasing products that are just good enough to convince buyers, and just bad enough to turn into service cash cows, is not unheard of in the software industry: anything to make a buck. This is not the fault of the free-software people. But their all-out focus on one single issue, "freedom" as they define it, and their refusal to attach any ethical value to another, quality, does nothing to help advance this critical component of the software profession's social and moral responsibility to the world around it.

8. HOLIER THAN THOU

Why do people write free software? The free software literature mostly cites one motive: an altruistic desire to help fellow human beings. Richard Stallman had this to say at the start of the GNU project:

> "*I'm looking for people for whom knowing they are helping humanity is as important as money.*" (www.gnu.org/gnu/initial-announcement.html.)

and recently (Linux World, 1999):

> "*Although we do business to make a living and live, there are things that are above and beyond that. Such as making the world a better place.*"

Wow!

GNU describes its goals (www.gnu.org/philosophy/pragmatic.html) as "*pragmatic idealism*":

> "*Every decision a person makes stems from the person's values and goals. People can have many different goals and values: fame, profit, love, survival, fun, and freedom, are just some of the goals that a good person might have. When the goal is to help others as well as oneself, we call that idealism.*
>
> *My work on free software is motivated by an idealistic goal: spreading freedom and cooperation.*"

The already cited Paris lecture opened up (no trace of any pandering to the audience) with:

> "*I can describe the idea of free software in three words: liberty, equality, fraternity.*"

The reader does not have to be a cynic to start wondering, after a few too many affirmations of one's unmitigated devotion to the sole purpose of "*helping humanity*", what the real story is. Saints do not typically go around clamoring "*I am a saint*". Tartuffe, however, did. (Molièrere's *Le Tartuffe, ou l'Imposteur* is required high-school reading in France. Maybe that's why French people often have such a hard time taking proclamations of absolute abnegation seriously.)

Questioning the motives of self-proclaimed benefactors of humanity is only normal, if only because they themselves not only question the motives of those whom they views as enemies of humanity (the commercial developers) but openly present them as liars:

> "*The economic argument [for charging for software] goes like this: 'I want to get rich' (usually described inaccurately as 'making a living')*" (www.gnu.org/philosophy/shouldbefree.html).

This slandering of commercial software developers, presenting the asserted desire to "*make a living*" as a lie covering their unabated greed, is pretty outrageous. It is all the more revolting, from an ethical standpoint, in view of the reality that many developers of "free" software of the taxpayer-funded category have the comfort of a monthly salary from an institution that cannot legally go bankrupt, such as a state-funded university. This provides a convenient vantage point for bashing commercial software developers, who have often put everything at risk to pursue their ideal. Many of them fail miserably, gaining only ruin and a divorce; some manage to "make a living"; a few will become rich, as they are indeed entitled to if their talent, effort, business sense and luck get them there. The functionaries have no right to deny them this right to live from honest work.

Here it should be pointed out that besides money a common motivating force for software entrepreneurs is their rejection by traditional institutions, including those who support free software. Many a successful industry figure has told the story of how he or she went to management with a great idea, only to be rebuked, and was so outraged by the reaction as to go away, implement the idea as a software package, and prove the idiots wrong. The motives of such people deserve respect, not abuse.

Even if all commercial software developers became wealthy, there would be no ethical basis for picturing them as greedy liars. Wanting to get rich is not morally reprehensible. If it were, shouldn't we hate people who buy lottery tickets? We don't. (Moralists tend to reserve their scorn for the lottery organizers.) At least the entrepreneur who starts a software company, perhaps with the hope of becoming rich, takes a personal risk, far higher than the price of a lottery ticket. That doesn't automatically make her a hero — and it is indeed one of the differences between commercial and "free" software developers that the former do not, in addition to other expected benefits, demand that the world hail them as saintly benefactors of humanity. But it also doesn't make her a despicable thug. There is nothing wrong about believing enough in one's ideas, and ideals, to put one's livelihood at stake.

The character assassination of commercial software developers by the extremists of free software, and their failure to accept that people with different views of the world can be honest and even idealistic in their own ways, has no moral basis.

It would be a natural reaction to counter at the same level, and dismiss the free software folks as a bunch of hypocrites. That would be a mistake. One should not respond to intolerance by intolerance. (And you never know: someone might, after all, advertise his sainthood and be a saint, if an immodest one[4].)

We should, as noted, question their motives; but this does not necessarily mean dismissing these motives as obvious lies. We must investigate them with an open mind, expecting that we will find cases of the advertised generosity — along with other impulses, some noble, some self-serving. We must, in other words, dispassionately try to understand, beyond the self-aggrandizement of some of the free software literature, why people do write free software.

[4] In the name of full disclosure, the reader must be warned that all comments about sainthood in this article are based on hearsay rather than on the author's personal experience or encounters.

9. REASONS FOR PRODUCING FREE SOFTWARE

People who stumble across free software for the first time are often surprised that anyone would willingly decide to give up any fees for the products he or she develops. But in fact it is not hard to find several reasons why, for many developers, this makes sense. All the reasons cited below occur quite commonly in practice.

According to some of the literature cited above, the sole reason is *desire to do good*. It is not the sole reason, but may be a reason. Someone sincerely wants to help the rest of the industry with software tools. There is no cause to doubt that this is often part of the free software developers' motives.

Another common motive is the realization that *there is no money to be made* from the development. You make your software free because you feel that you couldn't sell it anyway — there is no market, or you can't find investors to get a company started, or you realize you are better at programming than salesmanship. These days, for example, few people will pay for an editor; it is not surprising that many editors are free. Netscape, to take another example, only made its browser free (in two different ways: permitting use of the binary versions at no cost, and releasing the Mozilla product as open source) when Microsoft made its own Internet Explorer available on Windows at no cost, killing the market overnight. Until then, the Netscape browser was sold for a fee.

A motive that combines some of the previous two is the *attempt to correct limitations or deficiencies of existing tools*. You are not completely happy with an existing product (commercial or free) and fear that its producer won't correct the problem fast enough for your needs; so you decide to go ahead and do it yourself — either as an add-on to the existing product or as a redevelopment. You may feel, especially in the case of an add-on, that your contribution is not significant enough to convince people to buy the result.

Making a product free is also a great way to *enlist the help of others*. You realize that you can't do everything by yourself and discover that free, open-source development helps you find collaborators. One of the marvelous effects of the Internet has indeed been to enable software developers to put out announcements and obtain help from many people, often complete strangers until then, all over the world.

Another common reason is the *desire to learn*. You want to enter a new, promising area but don't know much about it. Because this is you first foray into that area, you realize that no one would pay you for developing the product, but that's OK because the primary benefit you expect is education. You realize that the best way to learn is often to do. Often this case implies the previous one as well: by developing the software in a free and open fashion you hope to get help from people who are also learning, or (better yet) already know the field.

Related to the desire to learn is *increasing one's marketability*. The industry is always in need of talented software engineers, but everyone who has to hire developers knows the incredible range in programming skills — most managers will readily admit that they see a ratio of 1 to 20 in the productivity of people with seemingly comparable backgrounds — and how difficult it is to evaluate candidates based on anything else than their tangible record of actually producing successful software. If you want to price yourself above your peers, what's better than spending a couple of years building a great free utility, then come to a technically savvy recruiter and say "Oh, by the way, I wrote the WIMP (Wimp Isn't Microsoft Project) Microsoft-Project look-alike for Linux, which 15,000 people download every day and Red Hat includes in its distribution". A great way to help the world, nibble away at the Great Satan of Redmond, and advance your career.

Also frequent is the *desire to make money*. This seems contradictory with the notion of free software, but is in fact quite compatible. At the time of writing, the stock market has just seen a successful IPO for Red Hat, a provider of Linux deliveries all based on free software. Anything Linux-based makes investors vibrate (almost as much as anything Internet-related). A number of companies, such as Cygnus (which according to the GNU pages has more than 50 employees) have built their business around distributing and servicing free software. Eric Raymond writes:

> "*I expect to be quite wealthy once the dust from the Linux IPOs has settled.*" (www.netaxs.com/~esr/travelrules.htm)

There is nothing wrong with this — except when commercial developers trying to "*make a living*" are accused of moral perversion because their alleged secret motive is ... to become wealthy. One can only compliment the founders of Cygnus and Red Hat: they saw in free software a great business opportunity, and banked on it. It is indeed true in some cases that you can make more money by giving away the software and charging for services. But it's an economic strategy, not an ethical matter. **A business model is not a moral imperative**.

This list of motives is not in order of importance, but if it were we would probably have to put pretty close to the top the *search for glory*. People want to get famous by writing a program that everyone will use. It is indeed interesting to see that many of the advocates of free software suffer no visible atrophy of the ego organ. An extreme case is Richard Stallman's well-known uneasiness at the widespread media attention devoted to Linux, a system that he feels (no doubt with some justification) could not have succeeded without all the groundwork of the GNU project. About a year ago, an editorial of the *Linux Journal* made much fun of a letter by Dr. Stallman requesting that Linux — and, as a result, the journal — be renamed to a more appropriate acronym, *Lignux*, acknowledging Linus Torvalds's debt to GNU. People in charge of both the operating system and the journal didn't seem to find the idea that brilliant. Judging from its current Web site, GNU seems to have given up on pushinng "Lignux", but continues to argue that the operating system should be called "GNU/Linux". We wish them luck. Like everyone else, they are entitled to recognition of their contribution. If they appear a little vainglorious at times, who would cast the first stone? Just cease slandering people whose aims are different. Some humans want to be rich more than they want to become famous; others want fame more than they want wealth. Most people, given the opportunity, would gladly be rich, and famous, and admired too. We're all humans. Just spare us the sermon on how much nobler you are.

This list of reasons why people go into free software is probably not complete. Its most striking feature is how close it is to what motivates the developers of proprietary software. The relative rankings may be different, but most of the categories will be present, to some extent, in both cases. This is not so surprising. Apart from a minority of saints, most people share the same basic desires, in a variable mix determined in part by personality and in part by circumstances. One of the true differences between people is tolerance, or lack thereof. Most of us can live with imperfect companions as long as we and they are tolerant of each other. Intolerant saints, real or self-proclaimed, are a dangerous proposition.

10. THE GREAT SATAN

Among the reasons for free software, one more deserves some consideration: hatred of the "evil empire" — Microsoft.

It is not hard to understand some of the reasons for the resentment of Microsoft. The company grew unbelievably quickly. Software engineers have a nagging feeling that it was partly a result of sheer luck: when IBM obtained DOS from Microsoft and hence created the modern Personal Computer industry, it didn't demand exclusivity, enabling the more prescient Microsoft to sell the same solution to every IBM competitor. (It is interesting to see that IBM's mistake was based on a reasoning — we don't make money out of software products, we make money from selling "iron", i.e. computers — that is not unlike the GNU/FSF argument that one should not sell software, but it's OK to sell computers, as well as services etc.) The reputation of Microsoft among developers is not helped by the persistent rumor about Bill Gates —hailed by the popular press as the all-time genius of software engineering — that when he did write code the result wasn't impressive. The perceived arrogance of Microsoft in dealing with partners, competitors, computer vendors and customers does not help.

Anti-Microsoft feeling is also reinforced by the nature of the Microsoft tools. Not only do they deserve their reputation of crashing frequently; even when they work, they often give computer-savvy users the impression of forcing them into an intellectual stranglehold. Instead of the Unix programmer's freedom to drive the interaction with versatile commands and scripts, you have to follow the exact scheme that the Redmond boys, in their wisdom, have devised for you.

For all that it is easy to miss the incredible contributions of Microsoft — and its de facto partner, Intel — to the just as incredible progress of the computer industry. By establishing a mass market that enabled staggering price reductions, "Wintel" has made the computer revolution possible. The most fanatical advocates of Linux do not seem to realize that, without Microsoft, Intel and the resulting 200 million compatible PCs, without the $500 400-MHz systems[5] including a complete operating system, there would be no Linux. The entire computer world, Microsoft groupies and Microsoft haters, is riding on the coattails of Microsoft.

It is easy enough to lambaste Bill Gates. It is less self-flattering for the anti-Microsoft community to understand why Microsoft has reached its unique status. There was the initial stroke of luck, of course. But there was also an in-depth understanding of the market. There was in particular a relentless effort at integration: you write a Microsoft Project plan, integrate a Microsoft Excel spreadsheet and a Microsoft Word document into it at the click of a mouse, post it on your Intranet with Microsoft Front Page for your colleagues to access with Microsoft Internet Explorer, have it automatically generate an e-mail with Microsoft Outlook whenever a deadline slips, and so on. The underlying engineering effort is huge, and does more to explain Microsoft's continued success than any conspiracy theory.

The other major explanation is the dismal failure of Microsoft's competitors. The torchbearer of commercial Unix, Sun Microsystems, had a once-in-a-lifetime opening circa 1989. Microsoft's nascent Windows operating system was late and buggy; the Sparc architecture with SunOS and OpenWindows was an exciting mass-market candidate. Instead Sun apparently decided that the mass market was not good enough because of low margin and ferocious competition, and that it was more profitable to sell high-priced servers — replacements for IBM mainframe — to Wall Street. During all that time Bill Gates (as was later reported) was scared of what Unix could do to his business; his competitors never seized the opportunity.

When they woke up, it was too late: Microsoft owned the desktop and, with it, people's minds. Microsoft even had become skillful enough to recover from its initial misreading of the Internet revolution. Its competitors had lost. Their feeble attempts to reinvent themselves with

[5] Post-publication note: these are supposed to be impressive figures (by 1999 standards). For the current equivalents, check the ads in last Sunday's papers.

other technologies (such as the Java programming language for Sun, a vain attempt to challenge the Microsoft power) will not reverse that historic defeat.

In achieving this result, Microsoft, however unpopular some of its tactics may be, relied for the most part on its business acumen and its technical dedication. The losers have no one to blame but the skills of their competitor, and their own myopia.

This applies in part to the free software community. The initial 1984 announcement of the GNU project (www.gnu.org/gnu/initial-announcement.html) stated that:

> "*Starting this Thanksgiving I am going to write a complete Unix-compatible software system called GNU (for Gnu's Not Unix), and give it away free to everyone who can use it. Contributions of time, money, programs and equipment are greatly needed.*
>
> *To begin with, GNU will be a kernel plus all the utilities needed to write and run C programs: editor, shell, C compiler, linker, assembler, and a few other things. After this we will add a text formatter, a YACC, an Empire game, a spreadsheet, and hundreds of other things. We hope to supply, eventually, everything useful that normally comes with a Unix system, and anything else useful, including on- line and hardcopy documentation.*"

The GNU project has produced many good tools, for which the software community must be grateful; but it has not produced, fifteen years later, the announced system. The closest thing to the kernel is Linux, which came from outside the project.

Microsoft envy is very palpable in the current excitement around Linux. On the floor of a conference such as Linux World (now held twice a year in the US, with equivalents in many other countries) one can hear the constant Microsoft bashing. The jokes can be funny up to a point, but soon one starts to realize that the problem most people have is not Microsoft. It's that they are not Microsoft.

A famous French comic strip and animated cartoon shows the Grand Vizir (who goes by the name Iznogoud) at the Caliph's court, constantly plotting to get rid of his debonair master, and constantly failing in his schemes, every new one cleverer than the last. His motto, repeated in every tone from envy to complaint to exasperation, is "*I want to be the Caliph in the Caliph's place!*".

Iznogoud is, pathetically, human. Most of us, given a choice, would really like to be the Caliph in the Caliph's place. What motivates the wheelers and dealers at Linux world may be, for a part, a desire to help their fellow computer users. But, mostly, they want to be the Caliph in the Caliph's place. They might just as well admit it in as many words.

11. THE ETHICS OF ACKNOWLEDGMENT

A point of ethics that arises in the design of free software has not yet been raised in this discussion (or anywhere else that I have seen).

Many free-software products are "copycat" versions of commercial software. The GNU project indeed made its mark by providing quality replacements for dozen of Unix utilities, from awk and yacc to troff and cc. (It is ironic that, even though the goal of the project, quoted above, was to replace Unix with a free operating system, most of its results were for a long time used mostly on commercial Unix systems.)

Such tools usually do not raise a *legal* issue: it is considered acceptable to start from a tool's specification and reimplement an equivalent version, as long as you don't use any of the original code.

The *ethical* picture is not as clear, especially given the uncompromising nature of the attacks on commercial software developers, abundantly illustrated earlier. By reimplementing someone's design, you are not stealing his code, but you are taking his ideas. That by itself is not unethical; building on the insights of others is in fact an integral part of the scientific process — at least as long as you respect them. Respect implies, in particular, *acknowledgment*.

Many open-source software developer seem to think they are somehow above this rule. Take the book *Gimp: the GNU Image Manipulation Program* by Michael J. Hammel, published in 1999 by SSC (publishers of Linux Journal). It includes a 4-page acknowledgment section, stating at the beginning that

> "*Of course, there are the original authors of the GIMP, Spencer Kimball and Peter Mattis. At the time of the GIMP's original release, they were undergraduate students at the University of California at Berkeley.* [...] *The story goes something like this: Spencer and Peter* [decided] *that a Photoshop-like tool for UNIX systems would be a fun thing to do.*"

GIMP is indeed a free-software "copycat" of Adobe's Photoshop commercial product. What's striking here is that nowhere do the four pages of acknowledgments include any suggestion that the people who designed Photoshop at Adobe might, possibly, deserve a modicum of recognition too.

This absence, unfortunately typical, shows a grave ethical lapse. That the Adobe developers were paid for their efforts does not remove the need to acknowledge their contribution. They invented a brilliant design and worked hard to implement it. However generous the authors of GIMP may have been to the world by producing a "free" imitation of that invention, they could not have done it without the anterior contribution of the Photoshop folks, and Adobe's money.

In software, much of the hard work and creativity goes into specifying a system. You also have to implement it, but that's not necessarily the place where you will show your brilliance. Seeing your specification, and having access to your system, a good developer can often reimplement it at reasonable effort. Doing something that has already been done by someone else — like Linux after Unix — is an order of magnitude easier, if only because there is an existence proof: the reimplementer knows it can be done.

It is both a boon and a bane that the interface of a product to the rest of the world, often the most difficult part of the work, is also the hardest to protect (since you cannot let customers use the product without telling them how to use it!). It's a boon for users, competitors, and free-software advocates; it's a bane if you work hard only to see a more powerful competitor prosper on the strength of your ideas. (The first spreadsheet program, VisiCalc, was an immediate hit, leading to great commercial success not for its inventor but for the next implementer, Lotus.)

It is legal, and it may be ethical, to start from someone else's design and reimplement it. It is profoundly unethical not to acknowledge it, whether the originator dealt in free or in commercial software.

This is not just a matter of being fair and courteous to the people who made possible your own job and hence your own little claims to fame and a great career. It is also a matter of acknowledging the tremendous contributions of commercial software. Commercial products not only save lives; more prosaically, they established the historical basis without which there wouldn't have been any free software. GNU, as we have seen, was unable by itself to build an operating system; the paragon of evil, Microsoft, provided the economic basis that made Linux possible. There would also not have been a GNU (and a GCC, a Linux, a Bison and so on) if there hadn't been a Unix in the first place, arising from the stroke of genius of two AT&T Bell laboratories researchers — Thompson and Ritchie — working in a strictly proprietary context.

Thompson and Ritchie, by the way, derived many of their ideas from Multics, a sixties' research project from MIT. The cycle is complete: commercial software builds on public software, and free software builds on commercial software. Sometimes you want one kind, sometimes the other. Both can be bad; both can be admirable. Both result from the hard work of skilled, hard-working, dedicated, often-idealistic software enthusiasts. It's the same people on both sides of the fence: the passionate "hackers" who think software, dream algorithms, eat data structures, drink objects, and long through the night will continue discussing the pros and cons of generating code for a machine with a clever instruction prefetch scheme.

Mongering hatred of the members of one of these two complementary communities is unproductive and unethical.

12. THE WORLD IS BIG ENOUGH FOR ALL OF US

What should be done? I will conclude with a suggested agenda for everyone, whether a commercial software developer or merely a computer user.

- Recognize the major contributions of the free software community, from Linux and GCC to TEX, LATEX and Ghostscript.
- Accept that both commercial and free software have a role to play, and that neither will ever go away.
- Be respectful of the authors of good free software.
- Try to convince them to apply the reciprocal goodwill.
- Refuse and refute the moral defamation of commercial software developers. If you are a software developer, be proud of your profession.
- Call the extremists' bluff by questioning their moral premises. Re-establish ethical priorities.
- Refuse the distortion of moral values and the use of free software as a pulpit from which to spread ethically dubious ideologies.
- Demand (in the spirit of faithful advertising) that the economic origin of "free" software be clearly stated, and that the products be classified as one among "donated", "taxpayer-funded" and the other categories described in this article.
- For Microsoft, whose unique position in the community creates unique responsibilities: promote a more open attitude towards the rest of the world; open up; be less mean. You can afford to be.
- For everyone: focus on quality. Work with every competent person — free or commercial software developers — to address the increasingly critical, and increasingly ethical, issue of software quality.
- Strive to combine the best of what the two communities have to offer.