The Ethics of Free Software

John Goerzen, `jgoerzen@complete.org`

December, 1998 (revi

It turns out that the difficulty encountered there was but one symptom of a serious problem affecting the very methods used to write the code responsible for running nine out of every ten of the world's microcomputers. The USS Yorktown failure is only one example of what can happen when trust is placed in inherently untrustworthy systems and design methodologies. Systems can crash on a daily or even hourly basis, [6, #reliability] with data loss a potential unfortunate reality when such events occur. [1] Critical systems can stop functioning, costing millions of dollars in lost revenue. In medical and space fields, for instance, lives could be lost.

The problem behind all this is *proprietary software*. Proprietary (or closed) software lacks many of the benefits that society has derived from the marvels of the industrial resolution. When a proprietary project is developed, there is no peer review. Imagine taking a flight on a jumbo jet designed by only a single person with no safety review from others. The thought rightly sends shivers up your spine. You're of course aware that humans are imperfect and mistakes do happen. Peer review is a key way to find the mistakes *before* they have serious consequences, such as an airplane crash, a bridge collapse, or the crippling of a warship.

Closed software also deprives computer scientists and developers from a chance to learn from those that have gone before. Imagine if Newton, for instance, didn't share his discoveries with others. Mathematics, physics, and scientific methods would certainly have taken a much longer time to develop, since many things in those fields have built upon Newton's ideas. [22]

Even though we have a long track record of the success of building upon previous innovations in virtually every field of science, individuals and organizations creating computer software and hardware conspicuously take the opposite view. Not only is this simply harmful to the quality and quantity of software and hardware available, it also is not the ethical thing to do.

2 The Free Software Solution

As we've seen above, there are tremendous problems with using and developing closed software. Fortunately, there is a solution: Free software (also known as Open Source[1]). When we talk about free software, we refer to freedom, not price; it's possible to charge for free software. Free software solves many of these problems by ensuring that not only can we build upon previous good software, but we also have an opportunity to have peer review of current software. This is often accomplished by licensing software under the GNU General Public License (GPL) [20] or a similar license.

Part II

A Utilitarian Analysis of Fr

3 Software Quality & Reliability

Software has a tremendously broad reach in today's society. Almost every person in the United States is affected by software, either directly or indirectly. Everything from surface mail and phone calls to airline flights and Internet-based commerce is handled by software. When the software behind these various activities is of higher quality, people that use this software benefit because it performs better and is more reliable.

They are happier because their airline flights are on time. This means people are less likely to miss an important meeting or a connecting flight at their destination. Internet orders are more likely to arrive correctly, meaning that the birthday gift for someone would be timely. To put it another way, poor software quality causes a great deal of harm. Errors in billing, credit reports, tax records, and the like are but a few examples of things attributed to software glitches. Any of these things can cause significant harm to people or companies. People can be unfairly targeted by collection agencies. They may waste money defending themselves against something they didn't do. Time is lost dealing with those people. A rejected loan, for example, could mean that a growing family can't get a larger home that they need. All of this can lead from low-quality software – and it does happen.

Software developers also benefit from better design methods. They can produce software in a shorter amount of time, and the computerized tools that assist them in development make their job easier.

Therefore, if it can be shown that free software produces higher-quality software, then it follows that we achieve greater productivity and more benefit for mankind, and free software is the ethical choice for software developers. Below, several attributes of free software are discussed, along with the reasons that they help to improve software quality.

3.1 Peer Review

One of the most important aspects in the design of any large and important project is peer review. In large projects such as operating systems, hundreds or even thousands of people work on that which ends up being millions of lines of code. A single missing character in that code can be enough to allow a security breach or cause a system crash. Humans are not perfect, and while computer scanning programs can catch some simple mistakes, there are many mistakes that can only be caught by another person looking at the code.

With closed software, in general, only employees of the company writing the program have access to its source code. This makes it impossible for others outside the company to find problems in the code *before* they cause damage. Worse, when there is a problem, the users of a program can do nothing to help find it.

With free software, such as Linux, the source is freely available for download. People are encouraged to look at it, to be critical, and to try to find bugs. And thousands of people do look at the code. The end result is that there are far more people proofreading and fixing code, and the program has fewer bugs and is more stable. [8]

3.2 Timely Fixes

When no source code is available, many bad things can happen, from security problems to devastating system crashes. The USS Yorktown is a great example. Do we really want to prevent the military, airlines, or any other industry in which lives could be at stake from being able to fix system problems in the field? If something breaks, they are denied the ability to fix it because they have no source code for the software! This isn't the only situation where vital government systems have been hampered by closed software. On December 15, 1998, a bug in Microsoft's closed mail server called Exchange took down two critical servers in the United States House of Representatives. Ironically, the bug occurred just when e-mail traffic was at its peak – days before a critical vote on the future of the President. House members, therefore, couldn't receive valuable feedback from their constituents. [4]

Both these situations were caused by bugs in proprietary code. In both situations, the people running the computers were completely helpless. They could not have fixed the problem regardless of how much effort they put into it; the lack of source code completely prevents that. A computer glitch is bad enough, but when people are denied the ability to fix it themselves, or to hire someone else to fix it in a timely manner, there can be very serious consequences. Lives aboard crippled ships could be lost.

problems, you are stuck with broken software, and there's nothing that can be done about it.

For software vital to the operation of businesses, such as billing and financial software, these consequences are plenty to demonstrate the ethical problems with keeping the source code to software a secret.

Free software presents a better alternative. When you find a bug in your software, you can fix it yourself instead of depending on the original vendor to fix it. Or, somebody else can be hired to fix the problem. This instantly takes a difficult, possibly insurmountable problem with software, down to something that could be fixed in a matter of hours or even minutes in many cases – a clear benefit of free software.

Another important aspect of this is that the users of a given free program can add their own features to it. If you're not satisfied with how something works, you can improve it! This capability is especially useful if the software vendor does not exist any longer or is unwilling to implement your desired changes. You get a better product, and the people using it are more productive.

The benefits are even greater than that, though. When people find bugs or add new features on their own to free software, they are encouraged to submit their changes back to the maintainers of the software. This means that every user of an free program is a potential contributor to it! What's more, if people have made modifications to a program – new features or changes to fit their taste – there is free peer review of these modifications! When people that add features cooperate with the process, which is almost always the case, the other people using the software can see the code and can spot problems. As a result, free software products can have bugs found and fixed faster, new features implemented sooner, and better reliability than closed-source products. [11] People that make changes for their own use can get these changes peer-reviewed, increasing the quality of their software. The result is that software is less likely to crash, systems are down less, and the software better meets the needs of those that use it – all of which contributes to an increase in utility.

3.4 Sec

Software developers can conceal security holes in their software, [10] intentionally or unintentionally. The end result is that people think that their data is secure when it really may not be. The consequences of this can result in loss of utility. A simple hypothetical will suffice: if the data on a courtroom computer holding privileged information about people's criminal records is leaked, people's careers and lives could be destroyed based on that information. Other cases of broken security could have a serious effect on national security, even causing loss of life in some situations.

When we have free software algorithms, we can be absolutely certain about their security level. This places the power to decide whether a given algorithm is appropriate in the hands of the users of it, not its author. The users of such algorithms can have a high degree of confidence that it is sufficient for their needs, thus increasing utility by decreasing not only the chances of compromise, but also extra efforts and worry needed to deal with it.

4 Legal and Monetary Issues

There are many different licenses that qualify as free software according to the Open Source Definition [2]. These include the BSD[3] license, the Artistic[4] license, the MIT/X Consortium[5] license, Netscape's Mozilla Public License[6], and the Free Software Foundation's GNU General Public License[7] [20]. Of these, I shall be discussing the GNU General Public License (GPL) in greatest detail. It is the most popular way to license software in the free software community, and is considered by many to be the best license under which to place software in order to ensure that it remains free forever.

In order to show that legal issues matter in a strict utilitarian evaluation, it has to be shown that these issues affect utility. With legal issues, this relationship can come in two forms: 1)

---

[3] htt

saving money on software and legal expenses and 2) saving time relating to legal issues.

If money can be saved, there is a benefit of free software. If a company is being discussed, the company's operation becomes more efficient. They can either spend the money they have saved on other things such as research, or pass the savings on to their customers. For individuals, the money can be used for other things – paying bills, household expenses, paying off a mortgage, etc.

Saving time also leads to benefits. This issue applies primarily to companies. If they save time dealing with legal issues, they become more efficient, leading to the same consequences as saving money. They can put more resources into development, for instance, which benefits not only themselves but everyone that uses their products.

4.1 The GNU General P

(and encouraged) to use it on as many computers as desired. For instance, if Compaq suddenly decided to pre-install Linux on the computers they sell instead of Windows, they would go from paying Microsoft $750 million per year [16, page 2] to pa˜2000˜ying a Linux CD vendor $7. While this is an extreme example, the general idea is there: businesses spend a lot of money for software.

There are indirect costs as well. Businesses must keep meticulous record of the software that they have, and the associated licenses, to make sure that they do not violate any licenses. If these records are not kept, or contain a single mistake, there is a risk that the business could face an expensive lawsuit. Keeping records takes time, and even if records are kept, there is still a possibility of a lawsuit.

Businesses must divert more time and energy from the things important to their company to meticulous record-keeping or perhaps even liability insurance. This means that either they must charge their customer more or they have fewer resources to devote to support, research, and development. This can lead to harm outside the business as well.

4.3 Individ

be correct, and he has a smaller chance of getting a nasty fine from the government.

5 Educational Value

In order to be a good programmer for something as complex as an operating system, years of experience with programming are generally required. One can learn the basic concepts behind programming, but this doesn't give much "real-world" experience. One tremendous benefit of free software is that every free software program instantly becomes a valuable educational tool.

One great way for programmers to learn is to look at good code from others. This is precisely the sort of advantage that free software provides. Programming students can study or even modify the internal workings of the Linux operating system as class projects, for instance. Hands-on experience with a modern operating system is a great way to learn skills.

If people are better-educated, we get higher-quality software, which of course leads to increased utility – a definite win.

Another benefit here is that students can use, at no cost, a free software operating system such as Debian GNU/Linux or FreeBSD at home. This gives them the same computing environment as they get at their place of learning, with the added benefit that they can tinker with the source code to every part of the system to their heart's content.

5.1 Learning Becomes F

bug in an e-mail client. Having the source to tinker with is an excellent opportunity to recruit people.

6 Possible Ob

However, this sort of argument ignores several issues. First, much of the most important or most common software for the computer, such as the operating system, is sold under a license that explicitly releases the software vendor from any such responsibility. In many cases, the vendor doesn't even guarantee that it's useful for what it's sold for. It would be difficult or impossible to collect any sort of money from such a vendor.

Secondly, this argument also ignores the fact that free software is less likely to have problems in the first place. If the software is less likely to have problems, there's a smaller chance that there is going to be any harm at all – a utilitarian win.

Third, there are various support companies that can provide guaranteed response time to problems with free software.

Finally, the argument ignores the legal time and cost involved with suing a software vendor. Sometimes this expense is so prohibitive that the hurt company will just take the loss and not try to sue anybody.

6.3 National Sec

wrong hands.

6.4 Programming

7.1 Apache

The Apache[9] free-software web browser is currently the world's most popular. 5

no farther than sendmail. Its authors have set up a profitable business that sells commercial support contracts and related items for sendmail.

7.3 Mozilla

On March

been described here. In many ways, Linux is bet˜2000˜ter than closed operating systems. Linux's stability is legendary, as is its speed. Linux has done a lot to help spread the word of the free software revolution. Programming students now have a stable platform to use for development. Businesses don't have to worry about crashes. People with slower computers don't have to worry about upgrading, in many cases.

In the eyes of some Linux advocates, however, Linux's greatest achievement has yet to be finished, but it is well on its way. Due to the extreme importance of the operating system on any modern computer, Linux is in a great position to demonstrate to the world the benefits of free software. Many feel that Linux could one day topple the closed-software paradigm and the huge companies that encourage it. Already, evidence of this is emerging. Forbes reports that RedHat Software, for instance, has projected revenues of \$10 million in 1998. [7] RedHat is almost exclusively selling free Linux software.

As we've seen above, the free software paradigm has many advantages. Linux is helping to bring the free software paradigm to the masses, and as such, is doing a tremendous favor for computing.

8 Conclusions

Given all of these ethical arguments and the tremendous amount of good that we've seen free software has made possible in our world, it's easy to see how free software makes sense from a utilitarian perspective. Most of the advantages of free software end up being in higher-quality software, lower cost, and smaller development time. All of these lead to tremendous benefits for mankind, and so I conclude that free software is a better paradigm, ethically, than proprietary software.

References

[1] Mike Andrews. Massive NT outage due to registry corruption. *ACM Committee on Computers and Pub*

[11] Eric S. Raymond. The cathedral and the bazaar. Revision 1.40, http://www.tuxedo.org/~esr/writings/cathedral-bazaar/.

[12] Eric S. Raymond. Closed source after 2000? http://www.opensource.org/y2k.html.

[1