

1 April 2003

Free software - can a simple idea bring ethics into the software industry?

The capitalistic society we live in has become so profit-oriented that the only products that matter, are the ones manufactured by large corporations. Profit has become more important than quality. Corporations such as Microsoft are producing new versions of software, just for the sake of making more and more money. In spite the fact that many users acknowledge that especially software that is developed by those firms is bloated, inefficient and does not serve the people but rather its creators, vast majority of people are still using it (Gaudin and Nash, 1998). To add insult to injury, the software licenses of the proprietary vendors usually include many limitations, which one must adhere to in order to be allowed to use the software. For example, a corporate vendor would never allow a user of its software, who is a competent programmer, to change the source code¹ of its software, in order to adapt it to his or her needs. Because such vendors treat their products as their intellectual property (which is quite understandable in the time of rule of the mighty dollar), they selfishly hide it from their competitors as well as their users. The movement started by a single man is trying to erase all such limitations and put things back in the right place as they were, before hardcore commercial software was born.

Richard Stallman, the self-proclaimed “last true hacker”, is probably one of the most controversial people in the field of computer programming. The ideals he believes in seem unimportant to many, but of vital importance to others. His movement will not solve the problem of worldwide famine or poverty; still, it does have an important impact on our society, though only in the area of software usage. His ideals are simple and age-old yet earth-shattering. He believes that all software should be free. Not free of charge, but rather free to use, modify and redistribute. In other words, he believes that software may be sold, as long as the source code is freely available ([Selling Free Software](#)).

¹Source code is textual form of the computer program. In this form it can be easily modified by a programmer in order to add features to the program or fix programming mistakes. Source code usually needs to be converted into binary code (using a compiler software) in order to be run on a computer. On the other hand, binary code is nearly useless to a programmer, because it cannot be easily modified.

In the early seventies Stallman became a part of a group of programmers at the MIT AI (Artificial Intelligence) laboratory, who called themselves “hackers”. Note that the term “hacker” did not always have the negative connotation that it has nowadays. The Jargon File defines the term hacker as “An expert at a particular program, or one who frequently does work using it or on it” (Raymond, 2002).

This community, which existed many years before Stallman joined it, believed that the information sharing is a positive act, which should be encouraged. They believed that all the source code should be free for all to use, study change and share. For a hacker sharing is more than just a privilege it is a moral obligation. In his or her belief no one should be denied access to the source code. The terms “intellectual property” and “software piracy, so often used by the big corporations when they decide to bring someone to court were unknown to hackers. They programmed (or rather hacked), because they took it as a fun and an educating thing to do. They did it for fame and glory, rather than money and their own well being. Unlike most of laboratories who treat their junior personnel in an under estimative way, any newcomer to MIT AI laboratory was more than welcome to work on any interesting project (Stallman, 1998). The main advantage of having the source code freely available is that a competent programmer can easily add features to a certain application and fix various problems it may have. Also, less code needs to be written from scratch, when one is able to use someone else’s code as a base of a new program.

In such a friendly environment, Richard Stallman was free to sharpen his already enviable programming skills. His first larger project was, interestingly, a text file editor, which he named Emacs (Editing MACroS) (Gross, 1999). However, this editor has nowadays become much more than just that. One can for example use it for reading e-mail, as a program development environment, playing games, such as Tic tac toe and much more. People often call it the Swiss army knife of the text editors. Stallman goes as far as to define Emacs not just as an editor, but also as a way of life, a religion. As a joke, he even founded a Church of Emacs with himself as its saint (Saint IGNUcius).

But Stallman did not stop at text editors. Along with other hackers of the laboratory he was also working on an operating system called the ITS, which stood for Incompatible Timesharing System. The Jargon File defines the term as “The foundation software of a machine; that which schedules tasks, allocates storage, and presents a default interface to the

user between applications” (Raymond, 2002). Microsoft Windows is also an example of an operating system. At the time, the ITS was considered a high-quality operating system, which made use of each and every feature of the hardware available at the AI laboratory.

The situation changed drastically in the beginning of the eighties. The type of computer, which Stallman was working with – the PDP-10 – was discontinued, as it became obsolete by other more powerful and cheaper computers. This made much of the software that Stallman and the team at the MIT created, obsolete as well, because it was incompatible with the new-generation machines. In particular, the ITS operating system, the greatest achievement of the community, was now but a mere toy insufficient for any serious work. To add insult to injury, most of the MIT AI laboratory personnel forgot about their ideals and went working for a corporate vendor, called Symbolics, which did not share the source code with others (Gross, 1999). By doing this, their work had become (in Stallman’s belief at least) unethical and immoral. Richard, an all time idealist, was thus left alone with nothing more but his ideals and an idea of a free (in sense of freedom) community (Stallman, 1998).

Having lost the community of hackers who shared their work with whoever wanted to make use of it, Richard Stallman decided to create his own. But first he needed a project with which to attract the attention of thousands of hackers who were lurking in the various newsgroups of those early times of computer networking. He decided to write his own operating system and he named it GNU, which stands for GNU’s not Unix². In 1985 Stallman wrote a message to a newsgroup titled “The GNU Manifesto” (Stallman, 1985) in which he wrote about the basic objectives of the GNU project. His ideology appealed to many and soon he had people all over the Internet helping him reach his goal. In the process, he also defined how the software developed in the GNU project should be treated. Four rules – freedoms he calls them - need to be adhered to, for the software to be accepted into GNU:

- The freedom to run the program, for any purpose.
- The freedom to study how the program works, and adapt it to your needs.
Access to the source code is a precondition for this.
- The freedom to redistribute copies so you can help your neighbour

² The word Unix is explained later on in this paper. As one can see the acronym GNU is recursive – it cannot be fully explained. The hacker community is very fond of this type of (“unexplainable”) humour and many unexplainable acronyms have appeared since.

- The freedom to improve the program, and release your improvements to the public, so that the whole community benefits (freedom 3). Access to the source code is a precondition for this. (Stallman, 1996)

In other words, anyone should be free to use, change, share and improve it. He named such software free software. Note that freedom is implied in the name, not price. Since the word “free” is very ambiguous, because it can mean freedom (as in free nation) or price (as in free drink), Stallman defines free software like this: “Free software” is a matter of liberty, not price. To understand the concept, you should think of “free” as in “free speech,” not as in “free beer.” (Stallman, 1996). He also believes that selling software is not immoral and should be encouraged, as long as the four freedoms defined above are not neglected.

Stallman had to make sure that the software developed within the scope of the GNU operating system is properly licensed. After seeing the collapse of the MIT hacker community, he certainly did not want to see his own in ruins. In order to prevent that, he also wrote the GNU General Public License. With this license he made sure, that he had a legal protection should anyone be denied access to any free software package or any of its versions released to public under the GPL.

Remembering how all of his work done at MIT was lost due to non-portable software (a software, which can only be used with a certain kind of computer and no other), Richard decided to use Unix as the base of his system. Unix is an operating system, developed by AT&T Bell Labs in the early seventies. All Unix-compatible operating systems need to be written in a programming language called simply C. By following a set of rules, which define the Unix operating system and using the C programming language for development, one is able to construct a “portable” operating system. This means that amount of source code, which needs to be rewritten in order to use a certain program on a computer that is completely different from the one that the program was written for, is severely minimized. Therefore, by choosing the C programming language and the Unix specification ([The Creation of the UNIX* Operating System](#)). Stallman prevented his software to be sentenced to oblivion, should the hardware that he used for development, become obsolete.

The first program that Stallman wrote was the GNU implementation of the Emacs editor. After that he and his team started working on a compiler (a program which converts source code into binary code – a form of program, which is used by the computer) for the C programming language, a set of tools for text processing, file finding and other tools. It is also curious, that not all software, which is used as a part of the GNU operating system, was actually developed by the GNU hackers. Some of the packages (which were licensed under the licenses Stallman considers as free) were taken from other projects and integrated into GNU. The most important two packages, which found their way into GNU this way, are the implementation of the TeX typesetting system named teTeX (which is primarily used for writing mathematical papers) and the implementation of the X Window System named XFree86 (a base on which graphical user interfaces can be built on).

Although the progress of the GNU operating system was - and still is – impressive, one very important piece of the puzzle to make a completely free operating system was still missing – the kernel. This is the most important part of an operating system, since it manages all the hardware in the computer and additionally keeping the applications, tools and programs that are in use at the same time from stepping on each other's toes.

Unfortunately the designers of the kernel, which is named Hurd ([The GNU Hurd project](#)), have stumbled into many obstacles, which essentially brought its development to almost a grinding halt. The Hurd kernel is still under development (as all software always is), but even after thirteen years of development it has not reached the stage where it could be used in production environment. According to Stallman, the only reason that Hurd is still being worked on is, that none of the developers want to throw all the work away.

One of the reasons that the Hurd kernel development process started in the nineties and not before is, that the microprocessors (the heart of the computer) available to the mass market were not powerful enough to run an operating system as complex and demanding as Unix. But in 1991, the prices of the microprocessors have dropped significantly and the legendary 80386 (or just 386 as they were often called) microprocessor made by the Intel Corporation became affordable to mere mortals. Thus more and more people were becoming interested in computers and what was once known to a handful of people was now available to the masses.

In such circumstances Linus Torvalds, a Finnish computer science student, who just bought himself a new 386-based computer was asking himself what interesting project could he start working on. Being dissatisfied with Minix, an operating system developed by a Dutch computer science professor Andrew Tannenbaum, which was developed as a teaching aid, he decided write a new and better kernel from scratch (History of Linux, version 2.1). Just as Stallman, he also sent an e-mail message to a newsgroup and just as it was the case with Stallman, the response he got from the hackers, who were just for a project to work on, was immense. In a matter of months Torvalds gathered around himself dozens of highly competent programmers who were communicating solely by e-mail. After a few initial versions he released the kernel under the GNU General Public License, for other people to be able to learn from it and accommodate it according to their needs and tastes.

Aside the processor prices and a bad design there is, in my opinion, one more reason why the Linux kernel succeeded and the Hurd did not. The Hurd kernel team took up a development process, which is widely practised by large corporations and is based on extensive and rigid testing before the software is released to public. The bad side of this model is, that although the testing is extensive, it is performed by a small group of people. Linus on the other hand believes that such development models pose many obstacles, by which one is diverged from development, because so many details need to be attended to, the most important of them being trying to think of all the cases that might crash a program or make it perform in a wrong way. His model bases on letting the whole community do all the testing that is needed, which means that the Linux kernel actually gets more testing (it is tested by hundreds of people) and that the developers do not have to waste their time with this cycle of development (Raymond, 2000).

Since his pet project was also to become his BSc paper he did not have the luxury of repeating the mistakes made by the team working on the Hurd kernel. Thus he chose a completely different design, which was much simpler and it made the kernel much more extendible, but at the time it was considered obsolete. Because of that he got involved in many arguments with Tannenbaum and his lecturer. But he believed that he was right and slowly but surely, he progressed. Unlike the Hurd, the Linux kernel succeeded.

Today, the Linux kernel bundled with the GNU and other software packages is used in many critical situations, such as the server machines, which provide various services, for example e-mail and web pages serving (the British Royal family uses for this purpose too). But the servers are not the only area of computing where the GNU/Linux is being used. The Yellow Cab Service Corporation uses it for coordination and navigation of their taxis, NASA and Boeing Inc. use it for simulation and research. Even the film industry uses GNU/Linux – mainly in the special effects area. It was used in the Titanic, Fifth Element, Interview with the Vampire, Lord of the Rings and many others (Lugos, 2002). Many software vendors got rich by giving the software away almost for free. That is, they provide their software in stores for people to buy, as well as on the Internet from where it can be freely downloaded. It seems that the only things they are making money on are printed manuals, certification programs and the support services, which they provide. The most known vendor of the GNU/Linux (Stallman insists on using GNU in the name to emphasize that the GNU software is an integral part of the operating system, although few vendors agree with his notion) called Red Hat, provider of the Red Hat Linux software package, is making profit for the fourth consecutive quarter ([Red Hat Reports Fiscal Fourth Quarter and Year End Results](#)). The fact that free software has succeeded in the corporate world too, proves that sharing the source code can still bring profit (Red Hat for example employs many of the programmers who work on the Linux kernel). It also proves all the software vendors, who claim that monopolism and not letting their users to have control over the software they bought is the only way to make profit, wrong.

In my opinion the question of ethics is an important issue, even more so in times when monopolism, hiding and lies is the preferred way of ruling the market. Success of the free software movement clearly is not questionable any more. It has gained a considerable user base. In comparison to other movements, the free software cannot just fade away, because it is not based on economy and profit, but rather on principles and ideas. Rich people like Bill Gates come and go, ideologists such as Richard Stallman stay and fight for their ideas. But on the other hand, the Free software movement will probably never be able to defeat corporations like Microsoft, at least not completely, because the way majority of the people think is simply too much self-oriented to be able to really grasp the idea that sharing and giving is important than just making billions and billions of dollars without any regard to ethics. Some have already learned that ethics can be combined with profit. They have also learned that ethics can actually help, because many people are prepared to solve problems even for free, as long as

they get something back, be it fame and glory, or simply a satisfaction that the problem is solved. Of course cannot live only on fame and glory. But numerous companies producing free software have clearly proven that one can feed one's family and still be able to respect the ethics of sharing laid out by Richard Stallman.

However unlike Stallman I do not believe that the ethics of the free software movement can be generalized on all the cases. Some software packages are simply into the free software economical model. An example of such a software package is Mathematica – a program designed for symbolic computing (*Key Elements of Mathematica*). The development of such a package requires very specialized knowledge and not that many people posses it. The market for such software is also relatively small in comparison to the operating system market. After all, an operating system (be it GNU/Linux, Microsoft Windows or any other) is the required piece of software, which makes a computer usable, whereas only the professional mathematicians use Mathematica. In cases such as Mathematica, the right to not disclose the source should in my opinion be preserved.

Though the idea of the free software is quite compelling and most of the software should eventually become free, certain types software are better to remain non-free. People like Stallman of course might not be satisfied with such a mentality. However, I do hope that the free software movement and the corporate world will be able to settle their differences and work together and thus bring the best of the both worlds into one.

Cited Works

Note: Because none of the books on the topic I am researching are available in Slovenia, I was forced to rely on the data provided on the web pages on the Internet.

Free Software Foundation Inc. The GNU Hurd project. 2003. 24 Mar. 2003

<<http://www.gnu.org/software/hurd/hurd.html>>

Gaudin, Sharon and Nash, Kim S. Computer users fight 'bloatware'. 12 Aug. 1998. 24 Mar. 2003

<<http://www.cnn.com/TECH/computing/9808/12/bloatware.idg/>>

Gross, Michael. Richard Stallman: High School Misfit, Symbol of Free Software, MacArthur-Certified Genius. 1999. 24 Mar. 2003

<<http://www.mgross.com/MoreThgsChng/interviews/stallman1.html>>

Hasan, Ragib. History of Linux, version 2.1. 24 Jul. 2002. 24 Mar. 2003

<<http://ragib.hypermart.net/linux/>>

The Jargon File version 4.3.3. Ed. Eric S. Raymond. 20 Sep. 2002. 24 Mar. 2003

<<http://www.catb.org/~esr/jargon/>>

Lucent Technologies. The Creation of the UNIX Operating System. 2002. 24 Mar. 2003

LUGOS. GNU/Linux v podjetjih. 16 Oct. 2002. 24 Mar. 2003

<<http://www.lugos.si/arhiv/podjetja.html>>

Red Hat Reports Fiscal Fourth Quarter and Year End Results

<http://www.redhat.com/about/presscenter/press/2002/press_q4fy2002/>

Stallman, Richard. The Free Software definition. 1996. 24 Mar. 2003

<<http://www.gnu.org/philosophy/free-sw.html>>

Stallman, Richard. The GNU Manifesto. 1985. 24 Mar. 2003

<<http://www.gnu.org/gnu/manifesto.html>>

Stallman, Richard. The GNU Project. 1998. 24 Mar. 2003

<<http://www.gnu.org/gnu/thegnuproject.html>>

Stallman, Richard. Saint IGNUcius. 2000. 24 Mar. 2003

<<http://www.stallman.org/saint.html>>

Stallman, Richard. Selling Free Software. 26 Aug. 2002. 24 Mar. 2003

<<http://www.gnu.org/philosophy/selling.html>>

Williams, Sam. Free as in Freedom, Richard Stallman's Crusade for Free Software. Mar. 2002. 24 Mar. 2003

<<http://www.oreilly.com/openbook/freedom/index.html>>

Wolfram Research. Key Elements of Mathematica. 2003. 30 Mar. 2003

<<http://www.wolfram.com/products/mathematica/benefits/>>

Raymond Eric S. The Cathedral and the Bazaar. 2000. 2003. 30 Mar. 2003

<<http://www.catb.org/~esr/writings/cathedral-bazaar/cathedral-bazaar/index.html>>