All Software Should Be Free Software

8th October 2004

Abstract

In the last quarter of century many changes happened in the fast world of computers. Usually these changes are about newer and faster hardware but in the middle of this contest another argument focused the attention of programmers and non programmers. This time the topic was on an ethical issue. Software should be *free* (libre) or *proprietary*?

1 Introduction

Have you ever tried to give someone else the last version of your best program? Including the source code? I did and I'd say it can be painful sometimes literally "donate" your own mind product and effort. Effort that, in many cases, is the result of several dozens or even hundreds of hard work hours. The only thing we should think when we do it is that the suffering is only a very small price to pay if we compare to what we have in return from the *free software community*[1].

Before starting we have to make a step back and define what is *free software*. Unfortunately because English language has a bug in it we have to start by defining terms. English has more words than any other language on the planet, it still has only one adjective for two very different concepts. In this paper when we are talking about free software, we are talking about free as in freedom, not necessarily price. So just think about free speech, NOT free beer.

Free software is a matter of the users' freedom to run, copy, distribute, study, change and improve the software. More precisely, it refers to four kinds of freedom, for the users of the software:

freedom 0 The freedom to run the program, for any purpose.

freedom 1 The freedom to study how the program works, and adapt it to your needs (access to the source code is a precondition for this).

freedom 2 The freedom to redistribute copies so you can help your neighbor.

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

Definition:

A program is *free software* if users have all of these freedoms [2].

Now that we defined exactly what free software means, we can explain who are the people that joined and join the community.Free software is a large community of users who run, share, copy and write free software. If we want to underline the date when this community was born, probably the best choice would be 5 Jan 1984. In that date, Richard M. Stallman[1] (or by the community usually referred as RMS) quit his job at MIT to begin developing a free software operating system: GNU (GNU is Not Unix). He was the first one, many others followed.

Speaking about what we have in return from the free software community, we have to underline two points:

1. The software we created available to everybody includes source code; the feedback we'll receive from other community programmers can obviously enhance and improve the product. Sometimes it happens that our idea might become the starting point of a very complex project to whom the entire community can contribute.

2. In exchange for our productive effort (though it still remains valid if we do nothing!), free software community give us the opportunity to download, use, redistribute and modify all software created year by year with the same philosophy. On the opposite side of the river there is proprietary software.

Definition:

A program is *proprietary software* if it is not *free software* or *semi-free* software. Its use, redistribution or modification is prohibited, or requires you to ask for permission, or is restricted so much that you effectively can't do it freely.

As we defined free software and proprietary software, there are a very large number of kinds in between identifying various software genders. Figure 1 tries to classify all the categories of software and their denominations.

We already defined free software so now we'll go on with other relevant categories:

- The term *open source software* is used by some people to mean more or less the same thing as free software.

[1] Richard Stallman is the founder of the GNU Project, launched in 1984 to develop the free software operating system GNU. The name "GNU" is a recursive acronym for "GNU's Not Unix". Stallman graduated from Harvard in 1974 with a BA in physics. During his college years, he also worked as a staff hacker at the MIT Artificial Intelligence Lab, learning operating system development by doing it. He wrote the first extensible Emacs text editor there in 1975. He also developed the AI technique of dependency-directed backtracking, also known as truth maintenance. In January 1984 he resigned from MIT to start the GNU project.

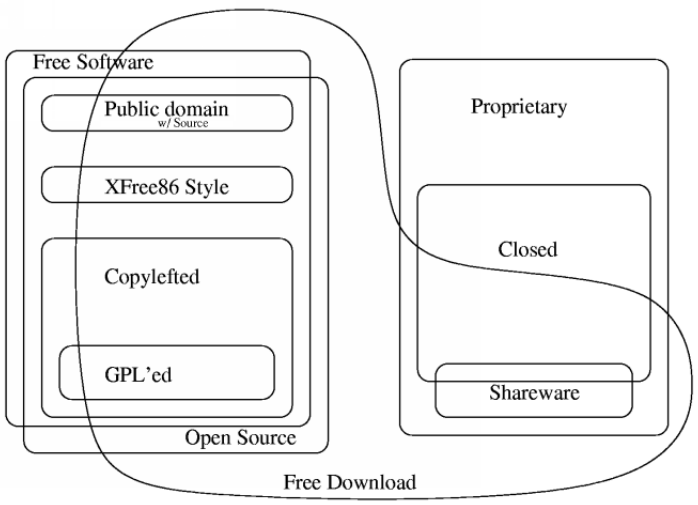

Figure 1: Software Categories

However, their criteria are somewhat lax; they accept some license restrictions that the free software philosophy consider too restrictive.

- *Public domain software* is software that is not copyrighted.
- *Copylefted software* is free software whose distribution terms do not let re-distributors add any additional restrictions when they redistribute or modify the software. This means that every copy of the software, even if it has been modified, must be free software (e.g GPL - GNU General Public License).
- *Semi-free software* is software that is not free, but comes with permission for individuals to use, copy, distribute, and modify (including distribution of modified versions) for non-profit purposes.
- *Private or custom software* is software developed for one user (typically an organization or company). That user keeps it and uses it, and does not release it to the public either as source code or as binaries.
- The term *freeware* has no clear accepted definition, but it is commonly used for packages which permit redistribution but not modification (and their source code is not available).
- *Shareware* is software which comes with permission for people to redistribute copies, but says that anyone who continues to use a copy is required to pay a license fee.

This paper will present its thesis that all software should be *free software* in the sense we just explained above. The Free Software Movement started in the mid '80s when the proprietary software became the largest way to produce software and simultaneously a bunch of researchers felt that copyright laws strictly applied on software were heavy slowing the progress. The paper will try to let the readers understand that free software philosophy [3] is the best way to create software of the highest quality and reliability. The largest proprietary software companies instead, push in the opposite direction because follow the guideline of profit.

Pargraph 2 presents the counter point of view. It illustrates mostly how owners (large companies) justify their power. The following one (paragraph 3) instead, presents *free software community* (our) point of view and how the battle against corporations is fight. The paper ends with a future perspective.

2 Counter argument

If we ask to big corporations like Microsoft, Oracle and Apple what do they think about Free software Foundation, Open Source Movement and in particular about GNU/Linux, they probably are going to give us the same answer, something that sounds like this:

> "Linux is a cancer that attaches itself in an intellectual property sense to everything it touches" [6].

They think, of course, in a completely different way compared to free software community. As owners of proprietary software they justify their power in many manners but their preferred argumentations could be recapped in to two points:

1. The emotional argument goes like this: "What I did is hard effort of my mind. It comes from me, so it's mine!"
2. The economic argument goes like this: "I want to get rich, and if you don't allow me to get rich by programming, then I won't program. Everyone else is like me, so nobody else will ever program. And then you'll be stuck with no programs at all!".

Proprietary software is sold with many restrictions we agree in the moment we buy the program and its license. Such restrictions are for example the unavailability of the source code but we do not have to underestimate that at the same time we cannot share the program we bought with our friends, neighbors and officemates and neither we can copy or change it (if we were able to do it!). The emotional argument is linked to all this restrictions, in fact the copyright law gives to the corporations the right to protect their products because they are considered *intellectual property*. As we will see in the next paragraph, the copyright should be in help for costumers but in the way it is used today it protects only the interest of the huge companies and their profits. Economists also believed that the free software model couldn't work in the free market. They were wrong in fact the GNU/Linux OS is a great example of success of the free software philosophy. The proprietary software model offers two key advantages. First, the ability to capture profits by restricting access to the fruits of their labor encourages software firms to make risky investments in the development and improvement of software. But the importance of the profit incentive goes beyond the creation of a particular software package. Most of the major software companies engage in research and development that may eventually spill over into how software is generally designed. Second, by owning the intellectual property rights, a for-profit company can control the development and subsequent evolution

of the product. While it may make mistakes in shepherding the project, it has profit motives to make the program as valuable as possible to consumers. Most software vendors, as Microsoft for example, have chosen to make sure that their products are compatible with previous versions, and to make sure that there is a "standard" current version. Having standard versions encourages the development of complements-for example, the vast array of applications software for Windows.

Proprietary software is an extremely successful business model. It has been very profitable, has attracted lots of investment, and has enabled software companies to create large numbers of well-paying jobs. The thing is that we don't believe that proprietary software is the right way to walk through. And neither their claims. The *emotional argument* falls very quickly when programmers have to decide to sign a contrat where they give all rights to a large company for a salary. In this way they loose their intellectual rights in behalf of the company. While the *economic argument* starts by comparing the social utility of a proprietary program with that of no program. As we saw free software exists and it is a strong alternative reality to proprietary software and that means there are people programming free software even if you decide to not program at all! Moreover the proprietary model has other weaknesses. First, to make profits proprietary corporation must limit access to the source code sufficiently to prevent copying. Most vendors of proprietary software choose not to make the source code available, even to customers. That makes it virtually impossible for customers to fix bugs they encounter or to customize software to their particular needs. This, of course, is mainly an issue for proprietary software used by information technology professionals who have the ability to modify source code. It is not an issue for most consumers, who have no desire to get under the hood. There is a second weakness. Although we generally think of free market firms as being quick to capitalize on profit opportunities, in the fast-paced world of information technology some customers may have needs that are not being met by software firms. The source code is not available hence it is impossible to add feature or fix bugs to a software I've already paid. Either we have to wait for the next upgrade/release or we have to ask to the vendor to develop the feature for us!

We strongly believe that free software is a valid alternative to the proprietary software. A taste of its power can be extracted from the past decade when in 1991 the operating system known as GNU/Linux was released under the terms of the GPL license. Born from the effort of the free software community and thanks to the code written by hundreds of free programmers around the world that believed in what they were doing. In the next paragraph we present the most important reasons we have faith in. These reasons are the backbone to understand why all software should be free software.

3 Free Software Philosophy

In the following part of the paper we'll try to explain why the society needs free software. Richard Stallman, the founder of the free software movement, believes that obstructing software by restrictions on the distribution and modification of the program cannot facilitate its use. Here we continue highlighting the problems caused by this kind of obstruction.

3.1 Impeding use of programs

A license fee is a significant disincentive to use of the program. That means that if a widely useful program is proprietary, a fewer people will use it. Whenever a user decides to pay for it, there is a so called zero-transfer sum of wealth between two parties. But each time someone decide not to use the program because he has to pay, this harms that person without benefitting anyone. In fact, either if the user pays or not, that doesn't reduce the amount of work to develop the program. Moreover, "only the largest proprietary software companies like Oracle and Microsoft actually can make a profit with licenses; most of the software companies make their profit from the typical model (see Figure 2) that we also use in the free software world: they make their profit from service contract, from support contract, from cost of improvement by a modification commissioned"[5].

The following analogy fits very well and it draws a clear picture of the risk of imposing fees over programs already developed. The analogy is about road construction. As everybody experienced, there are free roads (this time *free* is used with the *gratis* meaning) and toll roads. If we start constructing all new roads as toll roads and converting the existing free into toll ones, the result would be having tool booths at all streets corner! Such a system would provide a great incentive to improve roads but it would cause the users of any given road to pay for that road. However a toll booth is an artificial obstruction to smooth driving. Artificial because it is not a consequence of how roads or cars work. Moreover, in a poor country, tolls may make the roads unavailable to many citizens. Use of roads once built, should be free (of charge!). Of course, the contruction of a free road does cost money which the public must somehow pay.

3.2 Inability to adapt programs

One of the most important features of programs is the ease of modification. When we read a program written in an high level programming language we can understand what is the goal of the software and the way used to reach it. This feature is obviously one of the best advantages over the older technology. The thing is that today the most commercially available software isn't available for modification, even after you buy it. It's a black box, take it or leave it! Usually the source code of a proprietary program is kept secret by the owner and that means that only the program's owner can

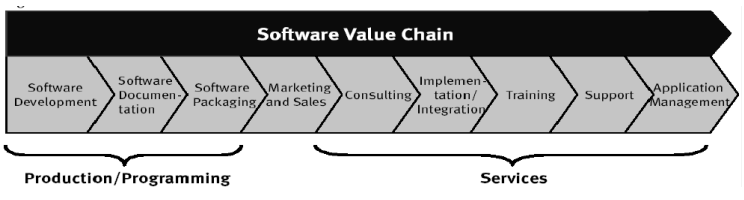

Figure 2: Software value chain

change the program. From another point of view if we don't have the source code but just the object program, for us is almost impossible to verify if the program does only what it should. For example (spyware) it could read information in my PC that I'd like to keep secret! Why shouldn't be allowed to check what the program does for real inside my PC?

3.3 Obstructing software development

Software evolves. It means that often programmers take an existing program and rewrite parts of it for one new feature, and then other people rewrite parts to add another feature. In some cases this process could last many years.The existence of owners block this very important process of evolution. Each time programmers have to start from zero when developping a software. Moreover source code of huge program is important for new programmers to understand how complex and different can be create big program. There is a considerable difference between small and large program and having access to source code helps in developping programming skills.

3.4 Software is a different mind product

Let's have a look at the *natural rights* argument. "I put my sweat, my heart, my soul into this program. It comes from me, it's mine!" [4]. You could consider right the previous sentence but several arguments can be mounted against the natural rights claim[7]. One is that it grants too much to the person who provided the labor, in fact a software writer never start entirely from scratch with no resource no help and no knowledge. The providers of these resources also have a claim to the resulting property. Another argument is that the natural right usually refers to the material aspects of property, the atoms. Bits, however, have different characteristics. Bits are diffusive and tend to leak. They can travel around the globe almost at the speed of light. They can be shared, replicated or copyed easily and cheaply. So they must be treated in a different way!

3.5 Copyright, CopyLeft and GPL

Unfortunately the Copyright law has being used for software in a bazaar way. Under the US Constitution, copyright exists to benefit users - those who read books, listen to music, watch movies, or run software - not for the sake of publishers or authors. The Constitution gives permission for a copyrighted system with this paragraph (Article I,Section 8):

> [Congress shall have the power] to promote the progress of science and the useful arts, by securing for limited time to authors and inventors the exclusive right to their respecting writing and discoveries.

and

> The sole interest of the United States and the primary object in conferring the [copyright] monopoly lie in the general benefits derived by the public from the labors of authors.

Stallman says[4] that copyright law was misinterpreted and in our opinion he is right. Hence he [and fsf] decided to use the copyright law with a new spirit. Usually copyright forbid to distribute, copy, share, divulge. He created the GPL - General Public License - where there is summarized the new way (CopyLeft) to use copyright law. Instead of using the law as in the other licenses, forbidding to do something, he used the law in the opposite way, by granting this right (all the 4 freedoms) and "forbid you to forbid those freedoms to anyone else" [5].

3.6 An analogy with the baby-milk powder

The effects of baby-milk powder on poor infants (which has sparked a Nestle campaign/boycott) provide an analogy to the effects of proprietary software. Sending information in Microsoft Word format to correspondents in Eritrea is analogous to Nestle advertising baby milk powder to Indian mothers. It encourages the recipients to go down a path which is not in their best interests, and from which it is not easy for them to recover. The apparent benefits (the doctor recommended it; we will be able to read the documents sent to us) may be considerable and the initial costs involved (to stop breast-feeding and switch to milk powder; to start using Microsoft Office) may be subsidized, hidden, or zero (with "piracy"), but the long-term effects are to make the recipients dependent on expensive recurrent inputs, and to burden them with ultimately very high costs.Moreover, because documents can be easily copied and because there are strong pressures to conform to group/majority standards in document formats, pushing individuals toward proprietary software and document formats can snowball to affect entire communities, not just the individuals initially involved.

Many other examples can be done in the controversy between free and proprietary software but we'll stop here, hoping that the best ones were already done and hoping that our point of view is pretty clear.

4 Conclusion

Bill Gates said:

> There's always been free software and it's always had a role to play in the marketplace. [...] The commercial platform have always done better than free platforms have.[8].

We know about the importance of free software, it grants that big companies haven't the monopoly. Microsoft knows about the danger that free software rapresents and in the past four year, as it saw that GNU/Linux advanced also as a desktop alternative, it has been studying how not to loose the big slice of marketplace conquered during the last two decades. It is not easy to make a prediction on what is going to happen next between these two deeply different way of living the software. We hope (believe) that free software movement is going to grow and win its battle against proprietary software.

References

[1] - The Free Software Foundation Community - http://www.gnu.org/software/gnue/community/community.html

[2] - The Free Software Definition - http://www.fsf.org/philosophy/free-sw.html [3] - Philosophy of the GNU Project - http://www.gnu.org/philosophy [4] - Free Software Free Society: Selected Essay of Richard M. Stallman 2002 fsf [5] - Bradley M. Kuhn's speech, Software Freedom and the GNU Generation, given at The Siebel Center for Computer Science at the University of Illinois in Urbana/Champaign, USA on 2004-04-24 [6] - Steve Ballmer - source: http://www.theregister.co.uk/2001/06/02/ballmer_linux_is_a_cancer [7] - Intellectual Property and Open Systems - Richard O. Mason - 2002 [8] - http://www.microsoft.com/billgates/speeches/2000/09-13malaysia.asp