Free Software = Software for Free?

Andrew Mikheyev and Laurens Vrijnsen

"Free as in 'free speech', not as in 'free beer"'
(Free Software Foundation[2])

In this paper, we present the basic characteristics of free software and open source software. We illustrate their impact on today's world of computing with a number of examples, before we compare the two against main-stream, proprietary software. The discussion will not be about holy versus evil, Redmond versus Red Hat, or Bill versus Linus; that has been covered by too many already. Rather, we will look at free software from a (system) developer's point of view: how about quality, security, usability, development speed, portability and profitability?

Introduction

Free software, open source... just a few terms that are emerging these days. Many associate it with software hacked together by a group of enthusiastic amateurs from that ancient UNIX-world. Software placed on the Internet to make it available for everyone, for free, without guarantees. That can't be serious software, can it? However... the growing popularity of Linux, also in the embedded world, is for a large part based on the fact that it is free software. But what does this concept mean?

In this article we will explore what "free" really means. First we introduce the concepts of free software and open source software. After a brief demonstration of its viability, we present a comparison of free software versus proprietary software. Finally we give some concluding observations.

Free software? Open source?

Contrary to what one may expect, the word "free" refers to freedom, not to "for free". Free software offers following freedoms to its users:

1. The freedom to run the program, for any purpose;
2. The freedom to redistribute copies[1] of both source code and binaries so you can help others;
3. The freedom to study how the program works, and adapt it to your needs;
4. The freedom to improve the program, and release your improvements to the public, so that the whole community benefits.

A program is free software if users have all of these freedoms. Thus, you should be free to redistribute copies, either with or without modifications, either for free or charging a fee for distribution, to anyone, anywhere. Being free to do these things means (among other things) that you do not have to ask or pay for permission.

Open source is another commonly used term to refer to this type of software. A closer look at the two different movements "free software" and "open source" learns that they use almost the same criteria for judging software, but with different rationale:

- the free software movement has an ideological focus towards freedom for the user;

[1] An exception is made if export regulations are violated by this, e.g. for encryption software.

- the open source movement takes a more practical approach: it promotes software reliability and quality by supporting independent peer review and rapid evolution of source code.

In this article, we will focus on the practical aspects of free software. Therefore, we permit ourselves to use free software and open source software as synonyms. Instead of having developers that create software in their ivory tower and then give it to their customers, open source software (and free software) creates a community of developers and users that interact.

The above-mentioned freedoms have two intriguing consequences for producers of software:

- they may not get a fee for every copy that is used;
- their solutions to problems (as found in the software) are exposed to everyone outside of the company, including their competitors.

Many vendors of proprietary software use copyrights and patents to prevent users from claiming the above-mentioned freedoms. On a more practical note, these tactics prevent knowledge and ideas to spread and be improved by others, thus limiting the speed of development and progress in the field of computing science.

In spite of these consequences, more and more companies are turning to producing open source software, as will be demonstrated in the next section.

A few examples of open source projects

One of the most well-know open source projects is the Linux operating system. The open nature of its development has boosted its development and therefore has created the basis for its current popularity, both with "hobbyists" and professionals. People who have problems with their Linux find that the community is not only open for development, but also for providing fast and good support.

Apache, by far the world's most popular web server with a 58% market share ([1]), gives another example of the high quality provided by open source software.

Apple was the first mainstream computer company to build its future around open source, and is partnering with the Apache Group, FreeBSD, NetBSD, and other open source developers to work on evolving the Mac OS X platform. It has released the core layers of Mac OS X Server as an open source BSD operating system called "Darwin".

IBM chose the open source Apache web server to support and bundle with its WebSphere suite. It has since released the Secure Mailer in open source and launched a web site to distribute alpha-status IBM technology in source, before they are licensed or integrated into products. This allows developers all over the world to both evaluate and influence IBM research and development.

Sleepycat Software builds, distributes, and supports Berkeley DB, an open source embedded database system. Their customers include many of the leading open source projects, as well as Fortune 500 companies whose own products are proprietary.

These are only a few examples. We encourage the reader to search for open source products that match his/her personal preferences... there is a good chance you will be pleasantly surprised by the results.

As shown in the above examples, "free software" can be commercial. One has to realize that software is more than just a collection of bits: it is a product that requires support to tune it to specific requirements. It is beyond the scope of this introduction to elaborate on possible business models here; we refer interested readers to the article *Open Source Business Models* by Anthony Liekens, later in this magazine.

Open source versus closed source

So free software can be economically viable, but what are the benefits? In order to answer this question, we will compare open source to its opposite, closed source or proprietary software, on a number of criteria:

- Quality;
- Security;
- Usability;
- Development speed;
- Portability;
- Profitability.

Quality

The good quality of the final product is a sum of two major components: good design and good implementation. For a major part of open source projects we can say that usually both are at a very high level. Two factors contribute to this - the accessibility of the source code and the professional level of the developers.

The source code being publicly available can be analyzed by thousands of amateur-programmers or professionals whose interests lie in the field for which the product is being targeted. Everyone is free to update the source code or send a feedback to the author of the erroneous module in the case a bug is spotted. This tremendously accelerates the testing procedure of the product in comparison with the closed-source projects, where the testing is usually done by a limited number of beta-testers, and only the project's development team does the corrections in the code.

The availability of the source code partly explains the high quality of open source products. However, this is not the only reason for that. The professional level of the developers participating in open source projects is on average very high. The Boston Consulting Group in one of its surveys partially mentioned in [4] found that the open source developers surveyed are mostly experienced professionals having on average 11 years of programming experience and the average age of 28.

Peer reviews play an important role in the open source development process and contribute to the high quality of the resulting products as well. Since all open source developers can see source code produced by the others, they can spot defects in the code and provide its author with feedback. If low quality of the source code becomes a persistent issue for some developer, then eventually he will have to leave the community.

All these factors contribute to the quality of the open source products, allowing them to score better in this category than the closed-source products.

Usability

As mentioned before, almost all open source projects are carried out by people who are fluent in modern software and hardware technologies. Traditionally, those people tend to concentrate more on the technical side of their work rather than paying attention to such details like user interface design. The implementation of a convenient user interaction in their products is not at the top of their priority lists. This is where the commercial closed-source products (usually working under Windows), definitely beat open source products with their amateur-like user interfaces.

Meanwhile, the open source community seems to have finally understood the problem. The situation with the usability of the open source software is constantly ameliorating. For already several years, newer versions of popular desktop managers for Linux having a constantly improving user interface are a good example of this positive trend. However, the developers still seem not to have found the right balance between the amount of functionality they offer in their interfaces and their ease of use.

Security

In the software systems there are many ways how security holes can appear. They can be caused by a bug that makes the system behave in a non-specified way. They can also appear as a side effect of some feature of the system - of which no one had ever thought before. The communication protocols used or implemented by the system can be poorly specified and use of them in an improper way can lead to security problems as well.

What will developers do in order to spot all the potential sources of security problems?

In the closed-source world testing is performed inside the company where the product is being developed. Some companies even hire professional hackers and let them explore the source code and the product itself to find as many potential security issues as possible.

In open source, all software developers of the world can have access to the source code of the open source products. If someone suddenly discovers a security problem, it will be known very soon by the open source community and the necessary measures will be taken by the authors of the system or concerned users. The fact that the source code of all widely used products is being constantly analyzed by thousands of software specialists all over the world raises the security of those products to the

level yet unreachable by the closed-source software industry.

Another advantage of open source is that using an open source product you can be sure that it doesn't contain any sort of back-doors - a hidden functionality that can be activated and used by the author of the system, intelligence or military organizations - without keeping you aware of this. As long as you have the source code of the system, it will be impossible to hide anything like this inside of it.

Development speed

The open source projects are usually developed by teams consisting of many people distributed all over the world. Most of them works on the project during their spare time, taking no obligations of any kind before the community. Some people do it because they believe source code should be open, others participate to improve their programming skills or just for fun. There are also people who do it for their professional needs, working on the parts that they need themselves. In all cases, the level of motivation of the developers is high enough to compete in development speed with the commercial closed-source projects.

Since Linux appeared in 1991, its today's releases contain tens of millions lines of code - all written by the participants taking no obligations of any kind before the community. Thus, Red Hat Linux 6.2 contained over 17 million lines of code, and Red Hat Linux 7.1 is composed of 30 million lines of code which is even more than those 29 million lines of Windows XP, which is considered to be the largest commercial project ever carried out! These figures are not only a testimonial of the high development rate that can be reached in the open source projects, but these figures also give us an evidence of a very high potential scalability of the open source development process.

However, open source development strategy has its drawbacks. The non- obligatory participation in the projects makes it possible for every participant to stop contributing whenever he wishes so. As a consequence, it is almost impossible to predict the release date for a next version of any open source product.

Another disadvantage of the open source development process is its development latency for supporting new hardware. One can run into troubles trying to install Linux on a brand new machine equipped with the latest graphics card, wireless connection card and other just released hardware equipment due to the lack of drivers for all this hardware.

Portability

Portability is becoming a very important concern for the developers who are working on the non-PC-based platforms. Embedded systems developers, for example, would greatly benefit from the possibility to tailor an external piece of software for their own hardware configuration. This is where open source solutions are much more attractive than the ones using closed-source ideology.

At present moment, many companies are working on their own versions of Linux for use in their proprietary embedded systems. This dispenses them from developing new operating system from scratch.

NetBSD operating system is just another good example of the portability of open source solutions. Up till now this operating system has been ported to as many as 48 different platforms! Different development teams got the possibility to port NetBSD to the platforms they are interested in, since its architecture and source code are publicly available for downloading.

Such an activity wouldn't be possible if the source code of the system had been proprietary and closed. The company-owner simply wouldn't have coped with the task of porting the system for so many hardware platforms. Most likely, it would favor one hardware configuration (one specific CPU) and produce builds for this particular device. This kind of strategy has been undertaken by Microsoft with their latest PocketPC 2002 operating system for which it had been announced that only Intel's StrongARM processors would be supported starting from that version.

Conclusions: applicability?

So strangely enough, free software seems to be most appropriate for those who are willing to pay for it. In the market of embedded software, it can lead to closer ties through co-development. Instead of sell-

ing software, companies can focus on selling support, e.g. tailoring software products to unique customer requirements. Open source software allows for fast progress in development of new software products by sharing new ideas. Exactly this is the secret to why free software products outdistance their commercial counterparts on a number of aspects.

For the large group of home users and office automation, open source software is becoming more and more attractive as an alternative for expensive software products. However, how a company can sell support to this group of customers remains unclear. Therefore, the viability of delivering open source products to this group is questionable, but companies must react to the competition offered by high-quality open source software products.

References

[1] Netcraft, "Netcraft Web Server Survey", http://www.netcraft.com/survey/

[2] Free Software Foundation, "Philosophy of the GNU Project", http://www.gnu.org/philosophy/

[3] The Open Source Initiative, http://www.opensource.org/

[4] Why Open Source Software? Look At The Numbers!, http://www.dwheeler.com/oss_fs_why.html/

Andrey Mikheyev holds an M.Sc. degree in mechanics and automated control received from French Graduate School of Mechanics and Microtechniques (Besanon, France). He also received an M.Sc. degree in computer science from State Power Engineering University (Ivanovo, Russia). He is an OOTI trainee since September 2001.

At this moment, apart from other professional interests, he is particularly interested in all products and technologies offered by Microsoft since he thinks that this passion will help him answer the ultimate question - "What does an IT-company and its employees need, to develop great products and thus survive and flourish on the today's hi-tech market?"

After a nine-month research project in Philips Research on a software architecture for the domain of emergency medical care, **Laurens Vrijnsen** received his Masters degree in Computer Science from the Eindhoven University of Technology in August 2001. Shortly afterwards he joined the OOTIprogram. His current fields of interest are software architecture methodologies and autonomous systems.

Laurens' experience with UNIX and free software dates from 1997, when he was introduced with the FreeBSD operating system. Ever since he has been a devoted worshiper of daemons and the UNIX design philosophy: creating small, reliable solutions.