24 Common Misconceptions about Open Source Software

Roland Stigge

September 15, 2007

24 Common Misconceptions about Open Source Software
by Roland Stigge

Published 15 September 2007

Contents

Abstract

This article gives an overview of common misconceptions regarding the Open Source phenomenon, touching several topics like cost, licensing, economics, scalability, community, warranty, support, quality, free software, distribution, compatibility, security, source code, profit, business strategy, volunteers, control, copyright and freedom.

1 Introduction

The following text clarifies common misconceptions about Open Source software. Basically, all of them are wrong and the respective following paragraphs explain why.

For an introduction to Open Source in general, please read the Open Source definition ["OSD"] and the Free Software definition ["FSD"], the latter with the Free Software Foundation's view on basically the same topic).

2 Misconceptions

2.1 "Open Source Software is gratis."

While Open Source software is considered as "free" in the "free speech" sense, it is not necessarily "free" in the "free beer" sense. As with proprietary software, there is a cost to the production. At this point, there is no difference to traditional development processes.

It is true that you can download many Open Source software packages for free, but there will be production cost to newly developed parts of the software. Don't expect Open Source developers to work for you gratis. You can just expect that when they already *have* some part of the software ready, they will possibly ship it under an Open Source license.

You can just hire Open Source developers to work on a special project to complete certain features. Technically, there is no difference to the traditional/proprietary software development process since there is a requirements definition, analysis, development/implementation, testing, deployment, even warranty as stated in the contract.

2.2 "You don't need a license for Open Source software. Open Source software is in the public domain."

While software which is in the public domain certainly qualifies as Open Source, most Open Source software is (currently) not in the public domain. It is actually copyrighted and your right to use, modify and (re-)distribute it comes from a license. Typical (and according to the Open Source Definition) officially approved Open Source Licenses can be found on the respective Open Source Initiative's page ["OSL"].

2.3 "Open Source software is the opposite of commercial software."

The opposite of Open Source software is proprietary software. Open Source software can be commercial, as seen with the big projects like Linux where millions of copies are sold on the server, desktop and embedded market. So there is no contradiction between Open Source and commercial software. (Although there might be a contradiction between actual proprietary and Open Source marketing strategies.)

2.4 "There is a difference [to the licenses and technical development] between 'Commercial' and 'Community' Open Source projects."

While different projects might have different characteristics in their development and marketing procedures (e.g., developed mainly in-house in a closed organization or admitting official committers from outside), their Open-source-ness is defined by the License which should be one of the OSI certified licenses for it to count as as Open Source project.

It is true that some commercial vendors provide "commercial" versions of their projects (or open-source effectively just a part of the project under an Open Source license) which are proprietary or at least not really Open Source according to the definition (license etc.). This can't seriously be considered as Open Source. In fact, it's only a way to fool (potential) clients into the Open Source hype. It still leads to vendor lock-in and all the problems that Open Source wanted to prevent in the first place.

It is also true that projects can be quite different in planning, development and day-to-day community involvement. But the outcome, a product that is shipped under an Open Source license that enables you to use, modify and redistribute the package, is always the same.

2.5 "There is no warranty to Open Source software."

While typical Open Source licenses include warranty disclaimers, it's possible to get warranty by contract with the software author or a separate support company. (And yes, this probably won't be gratis.)

The reason for the typical Open Source license to disclaim warranty is its freeness: Consider a consumer who would get warranty with the Open Source license. The license would allow him to redistribute the software package wherever he wants under the same license. The problem of warranty fulfillment (especially the legal and financial aspects, potentially to the rest of the world) is often not possible for the producer, even for large companies.

But by additional contracts, companies will sell you warranty. It is even normal that for contractual software development, the producer must provide a certain minimum warranty. This doesn't influence the regular Open Source license.

2.6 "There is no support available for Open Source software."

As with warranty, you can easily get a contract from software vendors and support companies. It is even typical for commercial Open Source companies to sell support contracts, which is an essential income source. Examples are the big Linux distribution vendors like Novell, RedHat and Canonical.

2.7 "Open Source has bad quality. It can't be good if it is gratis."

This is the typical attitude from the old times where cost and quality where directly related. This connection is completely gone with Open Source development models. It doesn't mean that good quality is an intrinsic feature of all Open Source projects, but there is some evidence to the tendency for good quality in famous Open Source projects: The bigger and more famous the project is the more distributions (Linux or other) include it and the more users will know it. These highly tested programs (like Linux and Apache) tend to be impressively stable, and when problems occur, those tend to be fixed within only a few days (sometimes, reportedly, within minutes!).

Open Source software can be (and in the widely distributed cases must be) developed in a peer review fashion, as known from science. This can't be achieved in typical proprietary environments where only a small number of people can review the source code.

Further, quality assurance measures from traditional software development processes can be applied to Open Source projects as well. And for the big Open Source projects like Linux, Perl, Debian etc., it is done for many years.

2.8 "There is a difference between Free Software and Open Source."

There are slight differences between the Free Software Definition ["FSD"] and the Open Source Definition ["OSD"]. While the Free Software Foundation focuses on simplicity of the definition (using, distributing, modifying and distributing changes), the Open Source Initiative elaborates further on political aspects like the prevention of discrimination of persons, groups or fields of endeavour.

But finally, the most important points to both views are the same. For a software to be considerd "Open Source" or "Free Software", it must be licensed under a certain license, e.g. GPL, LGPL, BSD or Artistic License. Since most software packages are licensed under one of these common licenses, they are automatically considerd both "Open Source" and "Free Software".

2.9 "Open Source is only a small part of the whole software industry."

This is a typical proprietary software centered view with measures from the respective business models. If you count the volume of sold software licenses, proprietary software will probably have the biggest share (with Open Source lacking accounting in terms of money and number of licenses). But since Open Source software licenses are typically not sold like proprietary ones, this kind of measurement can't be employed here.

The software market is actually highly penetrated by Open Source software, with Open Source products often being first choice because they are the "industry standard". Famous examples are BIND (DNS server), Apache (web server), and the Java, Perl and Python programming languages.

There is also no lack of Open Source projects for all kinds of tasks. At sourceforge.net, there are currently registered more than 155000 projects (not all of which are assumed to be active right now, but one can get an idea of the dimension). The Debian GNU/Linux distribution contains more than 11500 software packages which are maintained and usable right now.

Even more traditional software vendors embrace the Open Source development and distribution process. For example, MacOS X is partially based on Open Source (Darwin, Mach, BSD, etc.) and even Microsoft is in the process of registering their own Open Source licenses at the Open Source Initiative ["MSOSL"].

All this can be considered as some evidence that Open Source is not just a small part of this still fast growing industry.

2.10 "Open Source software is made by volunteers, being students or working in their spare time, they are not paid for their Open Source work"

In fact, in many Open Source projects, there is only a small percentage of committers who are completely volunteers, as seen in the Linux and Apache projects. ["LKD"]

2.11 "Open Source software comes from Universities or private entities."

In the early years (roughly before 2000), Universities, students and volunteers were heavily involved in Open Source projects. This changed during the last years. There are several reasons for this.

Universities realized the possibility of making profit of their works, mainly by patents, licenses, cooperations with commercial entities. Therefore, the pressure not to provide Open Source code from the results found in scientific processes increased. At the same time, project members left university and already knew how advantageous working with Open Source software can be and founded their own projects. Often done in start up companies like Thawte ["THWT"].

Now, the software industry actually drives a big share of the Open Source development itself as seen in project participant lists of the famous Open Source projects (see "LKD").

2.12 "You can buy Open Source licenses for using it per CPU"

Famous examples of companies who would like to give you this impression are vendors like RedHad, Novell and MontaVista. They sell "per-seat" licenses that seem to prevent you from using a big part of an Open Source operating system on more systems than you have payed for. Internally, most of these systems are Open Source software that you can actually use and share freely, just make sure not to be confused of a complicated system with added components that are not Open Source.

The real Open Source part of these products is licensed to be freely redistributable (and reusable).

2.13 "When we open-source our software, we don't have control of our copyright or we lose our copyright"

You still retain your copyright. Actually, with most Open Source licenses, you need your copyright to be able to assign it to the work under consideration. Then, you can provide the software under the protection of the respective license.

What you don't have is control about what people will do with your software. But with proprietary software you also won't have this privilege. Furthermore, you won't have control over redistribution of the software.

But your copyright stays assigned and keeps others from distributing the software under other licenses.

2.14 "The main point of the Open Source concept is the publication of the source code"

At this point, the name "Open Source" is a bit misleading, although not wrong. It emphasizes on the source code aspect. Other main aspects of Open Source licenses is the freedom to use, modify, distribute (even derivatives) of the software freely. There are even more political aspects to it, e.g. the prevention of discrimination of persons, groups or fields of endeavour (i.e. you can't restrict the software from being used in certain areas, e.g. warfare).

This means that the availability of source code is just one of several important points that must be fulfilled for a work being regarded as "Open Source".

2.15 "Open Source software is less secure."

This view is based on certain assumtions (which are probably wrong). First, there were many publicly announced security vulnerabilities for Open Source software in the past. This could be taken as a problem of openness: If you disclose your source code, problems (and vulnerabilities) can be found much more easily. This is correct. And it is considered a feature of Open Source workflows. It is the same "peer review" concept as known from sciences. This way, bugs and problems in the software can easily be identified and fixed, even without the original author or vendor. The latter is not possible in proprietary software. On the other hand, this doesn't mean that there are no bugs and (potential) vulnerabilities in proprietary software (or that those are not found and exploited). Vulnerabilities in proprietary software will also be disclosed, they won't also stay "proprietary".

A related typical idea from the traditional proprietary world is that if you keep your source code secret, it will be more secure since others can't see the problems with it. This concept is also known as "security by obscurity". And it is just wrong. It is just more difficult because software can be analyzed (reverse engineered, etc.) even without the source code.

Already some years ago, a famous security expert stated it like this:

> In the cryptography world, we consider Open Source necessary for good security; we have for decades.
>
> —Bruce Schneier, Crypto-Gram, September 15, 1999

2.16 "Open Source programs are not compatible."

While this concern is typically raised for certain applications and protocols in comparison of Open Source and proprietary versions, mostly the opposite is the case: Open Source has a tendency to obey open standards (or create such). The proprietary world has interest in locking customers into using certain products by certain companies to protect revenues. Therefore, they have no interest in making their software compatible by default, e.g. using open standards that other software (including Open Source software but also other competition) could use, or disclosing file formats and protocols.

2.17 "Open Source software is not ready for use in many areas."

While some years ago this was certainly the case where the user and developer base for general purpose Open Source systems was smaller, this is not necessarily true anymore, although this concern is repeated still.

Some Open Source examples for typical software packages:

Application Type	**Example Package(s)**
Operating System	GNU/Linux, FreeBSD, NetBSD, OpenBSD
Graphical User Interface	X.org, GNOME, KDE
Word processing	OpenOffice.org, LaTeX, DocBookXML
Audio	Audacity, Ardour, Hydrogen
Video	Cinelerra, MPlayer
Web Server	Apache
Web Browser	Firefox
Email Client	Thunderbird, Evolution
Graphics	Inkscape, Gimp, Dia

This is just a small current excerpt. For other applications, see the big Open Source software collections and sites like Sourceforge.net ["SFNET"], Freshmeat ["FRM"] and Debian ["DBN"].

2.18 "Making our software Open Source would decrease our profit."

This would be true if you base your revenue solely on software licenses. While this a problem for a few big companies in the market, this doesn't apply to many companies, since their core business is not copying and selling their own software but something else. Some examples follow.

If you are developing (and hopefully shipping!) a hardware product which contains software, it is not necessarily a problem for you to disclose the source code. You often don't need to change most of the Open Source products you are including (for free!), and the parts that you did change are adaptions to your special needs in your hardware. Sometimes this software will be driving the parts in the hardware that are a competitive advantage for you. But since you are not forced to also disclose the complete hardware wiring diagrams and design documents etc., the competition will probably not have it too easy to copy your product and take advantage of your investment. Remember that there are still the respective intellectual property laws like copyright and patents for your hardware. Also, don't forget that you are not automatically forced to disclose all the software components inside your hardware as Open Source. User interfaces and stand alone running software modules are common examples where you aren't forced to.

Another example is the case where you are basically using Open Source software. This is also true for people and companies that consider themselves as "developers". For example, if you are running some kind of analyst company running and developing software that supports you doing your job, your revenue comes from selling your expertise, the results of your research or other services. And it can stay this way, no matter if with proprietary or Open Source concepts.

Even typical software developing companies can open-source their software (like, for example Alfresco ["ALFR"]). The software will be developed and deployed in customer relations like in the tradi-

tional market. The Open Source concept isn't necessarily a problem here, but it keeps you well known in the community. Customers need specially adapted solutions. They need warranty contracts, support, training. They won't get it from the plain Open Source package. The best way to get it is from you!

2.19 "Open Source components can be integrated freely into a project."

While this can be true under certain circumstances, this can be applied only if your project follows certain preconditions. Of course, you need to obey the respective licenses. And if your project mainly consists of Open Source components and distribution channels, this won't generally be a problem.

But you can get into trouble if you mix proprietary and Open Source projects, linking your proprietary software with Open Source components where it is not allowed. For example, using GPL libraries (e.g. readline) enforces that the programs using the libraries need to have a license compatible with this license. You can either choose to open-source your respective program, too, or use other (Open Source) components instead (like, for example, LGPL'ed libraries).

Also, keep in mind that in some cases, Open Source licenses are incompatible, like for example GPL code linked to OpenSSL licensed code. Another example would be a pseudo Open Source license that prevents you from using your software in certain areas like commerce. The latter wouldn't be considered Open Source regarding the definition, but before using it make sure you can follow the license terms.

2.20 "Open Source is still a new concept to software development."

While the main part of Open Source software was certainly developed from the 1990s on, the concept reaches back to previous decades (even centuries). While Richard Stallman started to develop the GNU operating system (under the GPL) at the beginning of the 80s, Open Sources attitudes could be found at the very beginning of large scale computing, at the latest in about 1960.

Going proprietary wasn't a concept of software developers, and selling software as such wasn't a big business in the earlier decades of computing. Computers were the selling objects.

Open Source is a concept stolen from environments where peer review has always been an integral part of the workflow. See sciences. Actually, software can be considered a special kind of mathematics.

2.21 "If we open-source our software, there will be volunteers doing all the work."

This will only be true if volunteers are actually interested in your previous work and they can extend it while benefiting from it. Considering the already available software base in the Open Source world with software packages for all kinds of use, this is not necessarily the case if you have a special niche database product, or a graphics bitmap viewing software and now consider to open-source it. Although it might be advantageous for you anyway: People will see your expertise in a certain field and come back to you if you or your software could be helpful.

2.22 "The missing central control in Open Source projects make scaling to large project sizes impossible."

Actually, some of the biggest software projects are Open Source. For example, the Linux (kernel) and Debian Operating System projects include millions of lines of code, and multiple gigabytes of source code that interact successfully. Those projects have found ways to develop and progress. Certainly, this is even difficult with central control. Key points of large Open Source projects are distributed and networked development with massively parallel work. In the two given examples, there is a central repository available to get the software from, but actual development is not necessarily only checked into a central source code repository before the respective change is accepted. This can be done with separate subprojects, distributed branching (revision control systems like git, arch, darcs, bzr etc.) and derived projects (these so called "forks" are a kind of separately maintained branch).

2.23 "Open Source licenses are viral. If I incorporate Open Source code into my project, the whole project and derived works must be distributed under this license also."

First, not all Open Source licenses include the concept of forcing derived works to be licensed likewise. While the GPL forces derived works also to be distributed under the same license, other licenses like the

BSD license don't do this. It is a matter of choosing from which project (and the associated license) to draw code from.

In the case of GPL-like "viral" clauses, other code in the project under consideration may just not be suitable to be distributed under the GPL (e.g., a proprietary licensed part). Here, even the GPL can't force you to distribute the proprietary code under the GPL. But it can prevent you from conveying the resulting code (GPL licensed code together with proprietary) at all. You need to choose from different options. First, it is sometimes possible to approach the copyright holder(s) of the GPL licensed code to convey the code (additionally) under an alternative license which is compatible with proprietary code. Another option would be to license the proprietary code under the GPL which would often include a strategy change in the respective intellectual property department or asking the copyright holder in the case of external code contributions. Sometimes, a compromise with relicensing the whole project under a more "liberal" Open Source license, like BSD, is the only option left (except the prospect of not conveying the work at all).

Technically, there are different cases that either require likewise licensing of derived works or not. For example, linking tightly related code together will be a clear "derivation" in the GPL sense. But sometimes, code is coupled less. E.g., the Linux kernel is licensed under the GPL version 2, but it doesn't require programs that are run under the control of the kernel to be licensed under the GPL also. For kernel modules (mostly device drivers), it depends on internal details. They *can* be licensed under a non-GPL license but most often they are explicitely utilizing parts of the kernel that only allow GPL licensed code to make use of them. This way, kernel developers are trying to force device driver authors to disclose their source code under the GPL.

2.24 "Open Source is based on giving back. As users can use and distribute Open Source Programs, they need to send their improvements back and package maintainers will incorporate these changes."

Open Source basically ensures the user's rights to use, copy, modify and redistribute software. This doesn't necessarily include an obligation of the original author to incorporate other's changes back into the package. In certain areas, e.g. some commercial Open Source projects, common contributions are even categorically not accepted (here, at least not without copy disclaimers) ["RIEHLE07"].

While users are not expected to be thankful towards software authors for giving away their work for free, they also can't expect to have a strong voice in the development process. Often, patches and suggestions are highly appreciated by package maintainers whose only feedback is the community (in contrast to commercial customer demand). But one cannot rely on this principle. Just naturally, the convention of welcoming changes to Open Source project developed during the last decades without being forced upon developers and maintainers of software.

Users who have a certain demand for features and changes are free to maintain so-called "forks" or "branches" which are eventually incorporated back into the official main development line if the respective changes turn out to be useful. This is actually an important aspect of the development process of the Linux kernel, itself a large Open Source project. Some changes are incorporated only if they have been tested for a while in the public.

3 Conclusion

This article summarized some common misconceptions about Open Source software and explained some background about them. Although this list is certainly not complete, it includes topics like cost, licensing, economics, scalability, community, warranty, support, quality, free software, distribution, compatibility, security, source code, profit, business strategy, volunteers, control, copyright and freedom. While it didn't explain all those concepts in detail, it emphasized on how not to understand them in relation to Open Source.

4 References

[ALFR] *Alfresco Software, Inc.*, http://www.alfresco.com/

[DBN] *The Debian Project*, http://www.debian.org/

[FRM] *Freshmeat*, http://freshmeat.net/

[FSD] *Free Software Definition*, *http://www.fsf.org/licensing/essays/free-sw.html*

[FSF] *Free Software Foundation*, *http://fsf.org/*, *Home to the Free Software definition*

[LKD] Greg Kroah-Hartman, *Linux Kernel Development*, *https://ols2006.108.redhat.com/2007/-Reprints/kroah-hartman-Reprint.pdf*

[MSOSL] *Microsoft Open Source Licenses submitted to OSI, Microsoft Community License: http://www.crynwr.com/cgi-bin/ezmlm-cgi?3:mss:13324:200708:cokmgmoknbgepfbongjn and Microsoft Permissive License: http://www.crynwr.com/cgi-bin/ezmlm-cgi?3:mss:13323:200708:mkohfpmjekmjelobgffa*

[OSD] *Open Source Definition*, *http://opensource.org/docs/osd*

[OSI] *Open Source Initiative*, *http://opensource.org/*, *Home to the Open Source definition*

[OSL] *Open Source Licenses*, *http://opensource.org/licenses*

[RIEHLE07] *Dirk Riehle. "The Economic Motivation of Open Source Software: Stakeholder Perspectives." IEEE Computer, vol. 40, no. 4 (April 2007). Page 25-32.*

[SFNET] *Sourceforge.net*, *http://www.sourceforge.net/*

[THWT] *Thawte Inc.*, *http://www.thawte.com/*

I'd like to thank the members of the Open Source Research Network (http://www.osrenet.de/) who contributed important ideas to this article, namely: Lina Böcker, Robert A. Gehring, Christopher M. De Nicolo, Christopher Oezbek, Dirk Riehle and Jan Suhr. Further, all the unnamed people who discussed the respective issues with me during the last years.