Free, open or proprietary?

Philosophical differences in software licensing

Tom Chance

Software is a tool, a compilation of code that directs computer hardware, a program that empowers people to work more productively. Before Richard Stallman founded the GNU Project, many outside of hacker communities would have reasonably asked: why on earth is the ethics of software distribution philosophically interesting? But by formalising hacker conventions, Stallman kickstarted a revolution in the industry that now raises profound questions about areas of philosophical interest, most notably property. However, the precise differences between Stallman's conception of "free software", the term "open source" and the alternative: "proprietary" are often confused. This article seeks to disentangle the issues and present a clear analysis of each approach to software licensing.

Before I jump in, I ought to make clear a few ground rules. First, by "philosophy" I mean more than "thinking about something". By my use of the word, it would be nonsense to talk of "a philosophy of software engineering" when you really mean "an approach to software engineering". I want to go beyond the general beliefs and attitudes of each approach, beyond their techniques. It is true that, in terms of their techniques, free software and open source mean the same thing, and so identical philosophical issues will arise from their techniques. But the development methodologies (open sharing of code, many eyeballs finding and fixing bugs, etc.) were never an important part of Stallman's free software philosophy. So instead of looking at their techniques, I will analyse their orientation (or goal), their logic (why they adopt their particular orientation and techniques) and the limits of the space in which they apply.

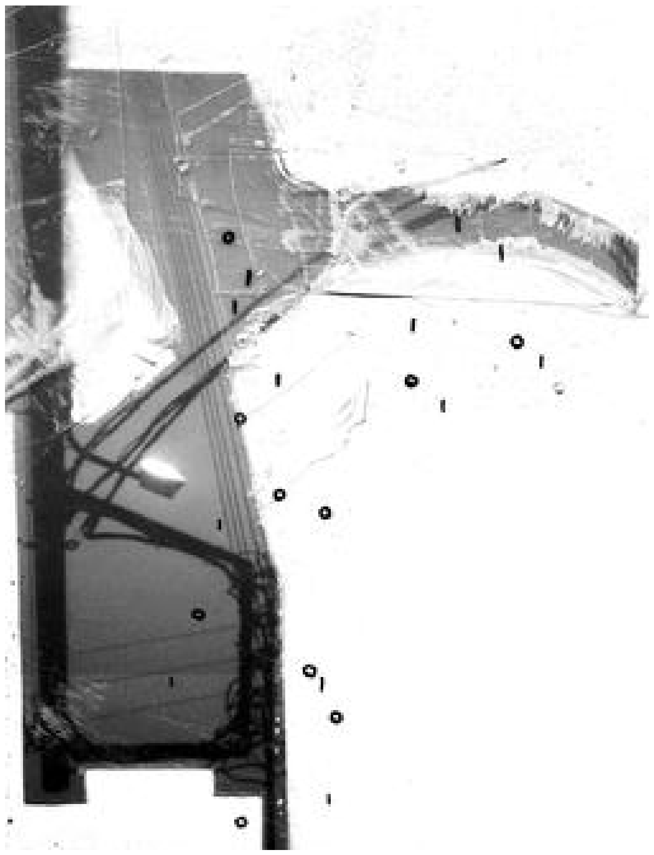
Free and open source approaches to software development may be identical, but their philosophies are radically different

Proprietary software

The orientation of proprietary software is to create good software. It's that simple. Its techniques, from a philosophical point of view, are similarly banal, involving various development methodologies and the application of copyright to both protect the software from outside interference and to protect the financial interests of the authors.

The logic behind this technique is, its proponents tell us, in the spirit of copyright: to reward the authors, and to promote future creativity. However, since propietary software may be released for free (freeware), the reward isn't necessary. Given that both the free and open source approaches also allow for rewards, we have to discount this as being philosophically distinct to the proprietary approach (though it is an open question for economists). Rather, the distinctive quality of proprietary software is that the source code is closed, making creation and modification the exclusive preserve of those to whom the owner gives access.

"data_cloud", by Campbell Orme. Released under the Creative Commons Attribution license



This is closely related to the logic of copyright, which isn't so clear. Traditionally property—including physical property—is defended in one of two ways: by reference to the inalienable right of the owner, or by reference to the benefits to society. In both cases, the crux of the argument is the justification of exclusion, such that the owner can exclude the public from using the property as he or she chooses.

For example, the English philosopher John Locke suggested that we have an inalienable right to own that which we have worked on. His argument was "not that the existence of private property serves the public good, but rather that rights of private property are among the rights that men bring with them into political society and for whose protection political society is set-up" (Waldon, 1998: 137). Locke went on to explain how we can come to own physical objects such as land that were previously in the commons. By mixing our labour with that which nobody owns, we come to own that part of commons (Locke, 2003). In the context of software, by mixing my work with ideas that I discover, I come to own the result; the commons can be thought of as containing undiscovered ideas, and ideas that are given by their creators to the commons, such as programming languages and techniques.

Locke suggested that we owned ourselves, and so we could own those objects that we mix our labour (ourselves) with

Of course in this context, Locke's argument faces a problem: in mixing my own ideas with those of others', I am not taking their ideas away in the same way that I might remove land from the commons by cultivating it and consequently claiming it as my own. This is because ideas aren't rivalrous or scarce, meaning that an infinite number of people could use the idea without reducing its utility. So there is no need for me to extend ownership over an idea if I can gain the same utility from it in the commons. Furthermore, because software builds upon the commons and upon ideas from other software, it is difficult to say in what sense you created a piece of software. If I write a classic "hello world" script in Python, should I be able to own that nugget of information held on my hard drive and exclude others from it? Should I be able to own the idea itself, excluding others from writing "hello world" scripts in Python? With large programs like OpenOffice.org it is slightly more clear that there is a significant amount of innovation and labour in the code, but the problem remains.

The impracticality of this technique suggests the necessity for a different logic, that of copyright as a bargain for the public good. According to this argument, the public benefits from the creation of software, but authors can develop software better if they can control access to the source code, whether for financial reasons or some other, and they can dictate the terms upon which the software is used and distributed.

The latter point, however, doesn't apply to freeware, where the author employs the techniques of proprietary software without seeking financial reward and without restricting the public's rights of redistribution. To explain this, one must explore one other aspect of the logic of proprietary software, that of producer and consumer.

The user passively consumes the software, and though it may well enable creativity, the software itself is an unchangeable commodity. This isn't true of any other kind of property; all physical objects are only limited in their mutability by the technical expertise of the owner. Therefore, uniquely the relationship between producer and consumer in this context allows no opportunities for productive community, be it hobbyist's clubs or businesses that will modify the software, except where the author steps outside of the proprietary norm and gives special access to the source code to particular individuals or groups.

Open source software

The orientation of open source software is described by the Open Source Initiative as producing good software. The definition of open source software is given in relation to proprietary software, comparing the techniques in terms of development methodologies and copyright licensing terms. It is the techniques that set the two approaches apart, not least because open source software rejects the main premise of proprietary software licensing—that it is better to restrict access to the source code. The logic for this difference, according to the OSI, is that "when programmers can read, redistribute, and modify the source code for a piece of software, the software evolves. People improve it, people adapt it, people fix bugs. And this can happen at a speed that, if one is used to the slow pace of conventional software development, seems astonishing." (OSI, 2004a)

This is achieved by subverting the traditional licensing of copyrighted works, specifically granting the right to use, modify and distribute the software to all who would take the opportunity. The logic behind this subversion is that it should result in better software. The open source approach therefore subscribes to the same philosophical justification of property in the software context as the proprietary approach, namely that being able to own the software serves the public good. If this seems paradoxical—that society's ability to modify the software depends upon the author(s) owning the software—it is because an author could release software into the public domain without the source code, and be under no compulsion to do so upon request, in effect releasing proprietary software. The technique of open source is subversive because it abuses the technique of proprietary software to renounce its logic.

Open source advocates rely on "economic self-interest arguments" without recourse to "moral crusades" and "ideological tub-thumping" (OSI, 2004d). In other words, open source as an approach explicitly avoids making itself philosophically distinct from proprietary software and any other intellectual property regime. Eric Raymond, a leading open source advocate, even tries to fit open source into Locke's approach to property. Locke suggested that property rights, based upon mixing one's labour with some part of the commons, only hold if the object of ownership is plentiful and promotes the public good. So Raymond says that, if we open the source code and forbid restrictions upon use, modification and distribution, we will increase the yield of useful work produced, and thus further the public good better than if we followed the proprietary approach. Restricting access to the program and its source code unreasonably abridges our access to a potentially infinite resource.

On these limited terms there can be no philosophical difference between the two approaches; both are based upon a particular application of copyright being the best way to produce good software. Even in the Open Source Definition, logic such as "no discrimination against persons or groups", which seems at odds with the logic and orientation of proprietary software, are explained in relation to their capacity to "to get the maximum benefit from the process", where a benefit is defined as the production of more good software (OSI, 2004b).

On community verses the producer-consumer model, open source advocates are a little more confusing. On the one hand, they claim that they are "promoting the Open Source Definition for the good of the community" (OSI, 2004a) and on the other hand they claim to promote the definition on "pragmatic, business-case grounds" (OSI, 2004c). As with the open source approach to property, this is because the community is recognised as the basis of the approach's pragmatic advantage over the proprietary approach. This has the interesting consequence that communities are only important if they contribute to the software, meaning that end-users who provide no input (whether it be code, documentation, money, etc.) are unimportant.

The Open Source Initiative maintains three central advocacy documents: one for hackers, one for customers, and one for businesses (both those producing and consuming software) (OSI 2004e; OSI 2004f; OSI 2004g). Their approach maintains the producer-consumer relationship, because the limits data_cloud_002, by Campbell Orme. Released under the Creative Commons Attribution license

of its space encompass only those that can contribute to the development process. Non-paying customers aren't stopped from moving into the "producer space" in the same way that proprietary licenses do, and they're not restricted in their use of the product, but neither are they afforded any more importance than customers of proprietary software.

To summarise, the open source "philosophy" is philosophically similar to the proprietary approach, because they both emphasise techniques that produce more high quality software. Their logic is subtlely different, their techniques radically so, and the limits of the space in which the open source approach operates are slightly wider. But their orientations, and therefore their overall approach to questions of property and community, are identical.

Free software

The orientation of free software is to create good software that provides certain socially useful freedoms. It is defined in terms of "liberty not price", a frame of reference entirely absent from both the proprietary and open source approaches. And crucially it is defined as an ethical orientation, not a pragmatic orientation (Stallman, 1992, 1994). According to the Free Software Foundation, the orientation is related to four kinds of freedom (FSF, 2004a):

- The freedom to run the program, for any purpose (freedom 0).
- The freedom to study how the program works, and adapt it to your needs (freedom 1). Access to the source code is a precondition for this.
- The freedom to redistribute copies so you can help your neighbour (freedom 2).
- 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.

Free software advocates reject the orientation and logic of both proprietary and open source approaches

The free software approach achieves this with exactly the same techniques as the open source approach: a range of software development methodologies based upon the free redistribution of the source code, made possible by the subversion of copyright through licensing. As Eric Raymond says, "the software, the technology, the developers, and even the licenses are essentially the same. The only thing that differs is the attitude" (Engel, 2004).

But the logic that connects these techniques to the "freedom orientation" is quite different to the open source approach. To begin with, the free software approach renounces the concept of software ownership. Software ownership is unethical, the leading figure of the free software movement Richard Stallman often declares. In contrast to Raymond's omission of natural property rights in his attempt to lend a Lockean justification to property, Stallman explicitly rejects this notion, though without reference to Locke or any other natural right theorists. Rather, in typical American style, he uses the position of the US constitution, which describes the copyright bargain, as precedent for his position. Property rights, he asserts, require a social justification and in the case of software there can be no such justification, therefore ownership of property is unethical (Stallman, 1992).

Though, as with open source, this logic concerns itself with socially justified property rights, Stallman judges the social justification upon grounds of the impact on the freedom of society rather than on the quality or quantity of software produced. He proceeds on a basis of comparative harm, asking if the harms resulting from the restrictions advocated by the proprietary approach outweigh the harms resulting from the freedoms advocated by the free software approach (Stallman, 1992), and advocates a kind of rule utilitarianism—a philosophical doctrine that creates ethical rules based upon maximising utility—that says: the (social) utility of always sharing software under a free license outweighs any harms, thus it is an ethical duty to always do so.

This ethical bias is also present in the free approach to the producer-consumer conflict. Stallman says that you "deserve to be able to cooperate openly and freely with other people who use software", and he encourages "the spirit of voluntary cooperation" (Stallman, 1994). One can apply these and other related ideas to any information-based work, where the hacker mantra that "information should be free" can overcome unethical restrictions. Quite how far this goes is unclear. One could advocate a limited form of the free approach and say that the limits of the space in which it holds only extend to the community of producers, as with the open source approach. Or one could extend the "spirit of cooperation" to empower consumers to engage with the producers in a way that can't be characterised as consuming.

"fields_and_fields" by Campbell Orme. Released under the Creative Commons Attribution license

Communities who customise or localise their software like KDE-Farsi and GNOME-Bengali are good examples of how this might take place, as are communities and cooperatives that are set-up to manage the spread of free software, such as in Venezuela and Brazil (Chance, 2004). Users, who with proprietary software would have simply consumed the software, are able to use it in a community context, and in so doing develop new communties and strengthen existing ones not based around software development, nor even necessarily software use (it may be an ancillary concern for the community). These activities are distinctive from the normal productive cycle that the open source approach endorses because they may often only benefit communities that are seperate from the wider "open source community"; to put it crudely, the free software approach advocates universal empowerment and liberation, whilst the open source approach endorses the good of the community in terms of software production.

In conclusion then, the free software approach is philosophically distinctive because, in contrast to both the proprietary and open source approaches, it is based on an ethical claim about the absolute importance of social utility, and about the relative social utility of different legal and development techniques. The approach rejects both natural rights and social bargain arguments for property in the software context, and subverts copyright law to create a global commons of software. Notions of community and cooperation are also central to the approach, both within the development community, amongst users, and between the two.

The two faces of the same animal?

Both the open source and free software approaches share the same techniques. Raymond was quite right when he said that the difference lay in their attitudes. The fact that most projects share contributors who hold either the free software or open source "philosophy" lends weight to the idea that the two approaches are just different faces on the same beast.

Whether or not you accept that conclusion depends on where your interests lie. Looking at it from the framework of either approach, it would seem that the important thing is to develop more software and, if you're a freedom person, to do so in a way that doesn't abridge our freedoms. From either perspective, their shared techniques achieve the goal admirably; the installation of GNU/Linux I'm using to write this article demonstrates as much. However, both approaches have attracted the attention of many a thinker from philosophy to economics, politics and law. For Marxists, the free software approach represents a critical challenge to property regimes; for some economists both approaches represent an experiment in gift economies; for many a political theorist they both present opportunities for democracy, freedom, you name it.

It's safe to say that because the term "open source" was coined to capitalise on free software's techniques in the business world, any thinker that leans upon the open source approach will be fairly content with less radical changes within the space they study. Management theorists, for example, can be content with applying the open development methodologies to their own previously heirarchical theories. Despite differences in orientation, many free software advocates are just as reformist. But some advance more radical critiques of contemporary approaches to property, community and the producer-consumer relationship, bouyed by the ethical basis of Stallman's position.

At the start of this article I admonished people for talking about their "software philosophy" when they actually meant their non-philosophical thinking about software. It should now be clear what the difference is, and why people confuse the two when looking at software production and licensing. Both approaches want to differentiate themselves from the proprietary approach; open source advocates refer to the set of techniques they advocate as the open source "philosophy" and free software advocates refer to the ethical orientation and logic they advocate as their "philosophy". The word "philosophy" is being used in a different sense each time, masking their actual philosophical similarities and differences. This is harmless semantics, a quibble from a philosopher, but underlying it are a range of questions that have been the bread and butter of philosophy for millennia. Long may they continue to plague our minds.

Bibliography

T. Chance (2004). In defense of free software, community, and cooperation (http://tom.acrewoods.net/writing/free-sw-community)

Creative Commons (Date unknown). Frequently Asked Questions (http://creativecommons.org/faq)

A. Engel (2004). Free as in Freedom - Part Two: New Linux (http://www.pressaction.com/news/weblog/full_article/engel12122004), Press Action Web Site

FSF (2004a). The Free Software Definition (http://www.fsf.org/philosophy/free-sw.html)

J. Locke (2003). "Second Treatise of Government, in Locke: Two Treatise of Government", ed. P. Laslett

OSI (2004a). Open Source Initiative, Web Site (http://www.opensource.org)

OSI (2004b). The Open Source Definition (http://www.opensource.org/docs/definition.php)

OSI (2004c). History of the OSI (http://www.opensource.org/docs/history.php)

OSI (2004d). Frequently Asked Questions (http://www.opensource.org/advocacy/faq.php)

OSI (2004e). Open Source Case for Business (http://www.opensource.org/advocacy/case_for_business.php)

OSI (2004f). The Open Source Case for Customers (http://www.opensource.org/advocacy/case_for_customers.php)

OSI (2004g). The Open Source Case for Hackers (http://www.opensource.org/advocacy/case_for_hackers.php)

E. Raymond (1999). “The Cathedral & The Bazaar: Musings On Linux and Open Source by an Accidental Revolutionary”

R. Stallman (2001). Free software: Freedom and Cooperation (http://www.fsf.org/events/rms-nyu-2001-transcript.txt)

R. Stallman (1992). Why Software Should Be Free (http://www.fsf.org/philosophy/shouldbefree.html)

R. Stallman (1994). Why Software Should Not Have Owners (http://www.fsf.org/philosophy/why-free.html)

J. Waldon (1988). “The Right to Private Property”

Copyright information

About the author

Tom Chance is a philosophy student, free software advocate and writer. He is the Project Lead of Remix Reading (http://www.remixreading.org), the UK’s first localised Creative Commons project, and is involved with a range of activist groups concerned with free culture. You can contact him via his web site (http://tom.acrewoods.net).