A microeconomic analysis of commercial open source software development

Date: November 7th 2007
Author: Mathieu Baudier (mbaudier@argeo.org)

Abstract

The particularity of open source software is how it is licensed. Anybody is free to use, modify and distribute free of charge software which is under an open source license. The only restriction is that the modified software has to be under the same license (and thus be available and modifiable free of charge as well). Open source software also typically comes without warranty.

This article aims at clarifying the impact of an open source license on the economics of software development. The goal is to provide theoretical material for pricing by suppliers of such software, and for cost/benefit analysis by private consumers or public providers of subsidies.

After briefly introducing the concept of open source software to a non-technical audience, we will first define the economic goods that we will consider:

- the availability of existing open source software
- the guarantee and the maintenance services attached to a software
- the development resources for the extension of existing software

We will then argue that the availability of an existing open source software is in practice a public good, and that there is therefore little incentive for a private supplier to provide it.

Then we will show that guarantee and maintenance services obviously are private goods, but also that development resources for software extension can as well be considered as private goods.

Finally, we will consider whether the specificities of software development (zero cost of physical production) and of the open source model (low cost of quality assurance) can compensate for a firm the intrinsic drawbacks of the open source model from a supplier point of view..

Introduction and assumptions

The Open Source Software (or Free Software[1]) concept was introduced in 1984 by Richard Stallman, a computer scientist at the Massachusetts Institute of Technology[2].
It is based on two facts well known from software engineers. First, it is much easier to understand a software system (particularly and crucially its errors), if you have access to the source code used to build it. Second, in some situations it is more efficient to fix yourself a software you rely upon, rather than waiting for its provider to do it.

Classical proprietary models in the software industry prevent both: the source code is secret because it is one of the main asset of the company, and your are not allowed to use modified versions of it which were not provided by the original company[3].

Stallman proposed a software licensing model whereby source code is available and anyone is allowed to modify and distribute it. Its decisive feature is not technical but legal: in order to prevent free-riders to improve an open source software and hide the improvements in order to gain competitive advantage[4], the various open source licenses specify that each modification should be licensed under the same (open source) license and thus be made public as well.

Open Source software greatly developed during the last ten years, the most famous examples being the GNU/Linux operating system (an alternative to Microsoft Windows or Sun Microsystems Solaris[5]) or the Apache web server (leader of the web server market since 1996[6]). These two systems are, like many others, developed by communities of volunteers receiving no other reward than recognition and deepening of their technical expertise.
In parallel, profit-seeking companies having business models completely based around the production or distribution of open source software have emerged (e.g. RedHat Inc. or JBoss Inc.), while classical software editors have started to release part of their software under open source licenses (e.g. IBM, Sun Microsystems).

There are lively debates on whether the open source model allows to produce better software (that is, with a lower rate of failure) at a lower cost (through the externalization of the quality assurance process to huge numbers of users, who got it for free and are incited to give a feedback). We won't address these questions here, and we won't try to compare the pro and cons of the proprietary and open source models.

We will just acknowledge that the open source model exists and is able to produce functioning software, usable for demanding applications.

For the purpose of this article we introduce the following assumptions or restrictions:

- **we will consider only production-quality software**: software that can be used for business-critical needs, and which provides a reasonable level of documentation and predictability in its release cycle
- **we will assume that proprietary and open source software compete in the same market**: for similar levels of features and support, consumers won't be biased towards one model or the other
- **we will only consider firms** producing open source software and/or providing services around them, not communities of volunteers, which have very different economic models

Definitions of the considered goods

The product and services related to software development can be split in three kind of goods. This division is the most important part of our argument (and the most debatable) since our conclusions will flow directly from it.

The **availability of the software** consists in providing an access to the executable code but also to its documentation, and in the case of open source software access to the history of changes in the code and to a system managing the bugs (failures) of the software.
The costs associated with producing this good (the availability) are not necessarily very high but they are not negligible. Machines and network bandwidth need to be provided, but also the time and technical expertise of those maintaining the various systems involved (web server, source control, bugs management) and the consistency of the whole (versioning, documentation ,etc.).

The **guarantee and maintenance services** are an insurance for the end user that any failure will be corrected with a high priority and in a predictable time frame.
They are important since all software systems have bugs and that open source licensed software come without warranty (producers don not want to be liable for losses caused by a software that anybody can get).

The **development resources for the extension of a software** are the time of the developers and the infrastructure required in order to add new features to a software.

The costs associated to both the maintenance (fix of bugs) and the development (introduction of new features) are the wages of the experts needed for the analysis, the development and the tests of the software and the costs of the related infrastructure (machines, maintenance of the systems supporting development).

The availability of an open source software is practically a public good

To determine whether a good is a public good we need to consider rival consumption and excludability:

- rival consumption: downloading the software and using it, does not prevent anybody else to download it or use it.
- excludability: because of the very nature of an open source license, nobody can legally be excluded from downloading and using the software

The non-excludability aspect is the most debatable. It is of course technically possible to prevent somebody from downloading the software. But, in that case, the software would loose its open source nature as clearly stated in Section 5 of the Open Source Definition (see Annex A): “The license must not discriminate against any person or group of persons.” or in the Free Software Definition[7]: “The freedom to run the program means the freedom for any kind of person or organization to use it on any kind of computer system, for any kind of overall job and purpose”.

Just as for any non-excludable good, there can be free-riders, which is what usually puzzles people most, especially in the IT industry which is used to a stringent protection of its software and related source code.

The non-rivalry could also be discussed since the network bandwidth used for download is limited

and a rival resource. Practically though, there are never enough people downloading at the same time, and for the most famous software, “mirror” servers duplicating the code across the world are provided.

Whether the availability is a pure public good is highly debatable, but what interests us here is that the practical consequences are the same: whereas there are clear benefits for the users to be able to legally access and use the software, there is no incentive for a firm to provide it[8].

Here we only analyzed the availability of the software, not its actual production, that we will consider later as the addition of new features. We wanted to isolate the puzzling aspect of the open source model. As we will see it is actually pretty limited.

Guarantee and maintenance services are typical private goods

Guarantee and maintenances services are both excludable and rival consumable:

- excludability: a firm can decide whether or not to provide these goods to a given consumer
- rival consumption: the resources used by a firm to provide these goods for a given consumer, such as the time of its expert or the computing power of its machines, cannot be used for other consumers

These services are therefore private goods and can be traded on a market, along with similar proprietary products. For these goods, the open source model has not fundamental impact, except that other firms can also provide a similar service related to the same software even if they don't produce it.

This is actually the main source of revenues for a lot of open source companies, such as MySQL AB, which produces the most widely used open source database system: “MySQL Enterprise includes 24x7 Production Support that helps you deliver continuous availability for your production database applications.”[9]

The resources for the extension of existing software are private goods as well

Consumers of IT systems are often interested in features which would increase the efficiency their system bring, but which are not yet available and need to be developed. A consumer already using (or free to use) existing open source software can therefore value the addition of new features.

The resources required for these extensions are private goods for the same reasons as the guarantee and maintenance services: consumers can be excluded and resources need to be shared. Just as for guarantee and maintenance, in the open source case even firms which are not producing the software can compete as well.

But here the open source model has more impact, since the addition of new features actually consists of the production of software whose availability is practically a public good, as previously discussed.

First, there is the question of the birth of the software itself. Only what already exists can be extended. In most cases, the initial development will be an upward investment of the firm producing the software (e.g. a financial investment, or the time spent on an university project as is frequently the case).

Then, and more crucially, the fact that the availability of the improved software will practically be a public good reduces the value of the software for the consumer because, due to the free rider problem, it will not gain any competitive advantage with it. The value will only be related to the ability to drive the development that best fits its need and to the guarantee of a delivery schedule[10].

However it can be argued that the same is true (but to a lesser extent) with a classical proprietary model, since competitors of the consumer can also buy the improved software. The only solution to completely enjoy a competitive advantage is for the consumer to keep extensions private, but then the benefits of the pooling of the code are lost.

Conclusion

Two main points should be taken into account by a firm considering to supply open source software.

First, **the availability of the software is practically a public good and there is no revenue to be directly earn from it**.

However, an indirect benefit is the prestige in the open source community. This can be valuable since the provided software will probably itself rely on other open source components and being itself part of the community can prioritize its needs of support for or extension of these components. This is especially true for community based projects.
Another benefit is the added credibility toward consumers for the supply of the related private goods (maintenance and extension) for which there are competitors: the main maintainer of the software is the most credible[11].
From another point of view, providing a public good can also be used as an argument for requiring government subsidies.

Since the costs of supplying availability are fairly low, due to the characteristics of software development (zero physical production costs), these benefits may outweigh them.

Second, **developments for the extension of the open source software will be less valued by the consumers**, because they won't gain any competitive advantage with them.

Suppliers of extension developments should therefore focus on customer needs where tight schedule and specificity of the needs are more important than the competitive advantage that could be brought by the use of the new features.
Another (and complementary) approach is to offer, on top on the open source basis, highly specific developments (of no use for other customers) which would remain private. For such an approach the use of business-friendly open source licenses would be recommended[12].

The commercial activities related to open source software development have some particularities but they are not fundamentally different from those related to proprietary software.

These particularities intrinsically leads to lower commercial margins (providing a public good) and/ or lower demand (less attractive offer for extensions developments). A rational supplier should therefore make sure that he makes the best use of the specific benefits of open source in order to outweigh these hindrances.

Annex A: The Open Source Definition

Source: http://www.opensource.org/docs/osd.php[13] (accessed November 7th 2007)
Submitted by Ken Coar on Fri, 2006-07-07 15:49.

Introduction
Open source doesn't just mean access to the source code. The distribution terms of open-source software must comply with the following criteria:

1. Free Redistribution
The license shall not restrict any party from selling or giving away the software as a component of an aggregate software distribution containing programs from several different sources. The license shall not require a royalty or other fee for such sale.

2. Source Code
The program must include source code, and must allow distribution in source code as well as compiled form. Where some form of a product is not distributed with source code, there must be a well-publicized means of obtaining the source code for no more than a reasonable reproduction cost preferably, downloading via the Internet without charge. The source code must be the preferred form in which a programmer would modify the program. Deliberately obfuscated source code is not allowed. Intermediate forms such as the output of a preprocessor or translator are not allowed.

3. Derived Works
The license must allow modifications and derived works, and must allow them to be distributed under the same terms as the license of the original software.

4. Integrity of The Author's Source Code
The license may restrict source-code from being distributed in modified form only if the license allows the distribution of "patch files" with the source code for the purpose of modifying the program at build time. The license must explicitly permit distribution of software built from modified source code. The license may require derived works to carry a different name or version number from the original software.

5. No Discrimination Against Persons or Groups
The license must not discriminate against any person or group of persons.

6. No Discrimination Against Fields of Endeavor
The license must not restrict anyone from making use of the program in a specific field of endeavor. For example, it may not restrict the program from being used in a business, or from being used for genetic research.

7. Distribution of License
The rights attached to the program must apply to all to whom the program is redistributed without the need for execution of an additional license by those parties.

8. License Must Not Be Specific to a Product
The rights attached to the program must not depend on the program's being part of a particular software distribution. If the program is extracted from that distribution and used or distributed within the terms of the program's license, all parties to whom the program is redistributed should have the same rights as those that are granted in conjunction with the original software distribution.

9. License Must Not Restrict Other Software
The license must not place restrictions on other software that is distributed along with the licensed software. For example, the license must not insist that all other programs distributed on the same medium must be open-source software.

10. License Must Be Technology-Neutral
No provision of the license may be predicated on any individual technology or style of interface.

1 The original (and still relevant) term is Free Software with free being meant “as in free speech, not as in free beer”, and also contains notions of principle. The Open Source Software denomination was introduced in 1998 by some who wanted to make the concept more “salable” to business people, and is rejected by Richard Stallman and the Free Software Foundation. Technically and legally the two concepts are strictly equivalent, and we use the Open Source denomination here for the purpose of clarity, with no implication on our own opinion in this debate.

2 Richard Stallman, “The GNU Project”, http://www.gnu.org/gnu/thegnuproject.html (accessed November 7th 2007), originally published as Richard Stallman, “The GNU Operating System and the Free Software Movement” in Open Sources, eds. Chris DiBona, Sam Ockman, Mark Stone (Sebastopol, CA: O'Reilly and Associates, 1999)

3 This is typically why many big organizations (such as big banks) prefer to develop their critical IT systems in-house rather than relying on an external supplier, although the benefit of specialization and pooling of experience are lost.

4 The most famous example is the DOS operating system which was freely available in the 80's but not protected by such a license. Microsoft adapted it to a new concept of hardware produced by IBM, the PC, in order to win a tender. This slightly modified version became MS-DOS and later on the Windows operating system, which now enjoys a near monopoly on its market.

5 The core of the Solaris operating system was actually released under an open source license by Sun in 2005, but is not production capable as of November 2007. Sun press release: “Sun Announces Open Source License for Solaris Operating System”, http://www.sun.com/smi/Press/sunflash/2005-01/sunflash.20050125.1.xml (accessed November 7th 2007)

6 “Netcraft web server survey of October 2007”, http://news.netcraft.com/archives/2007/10/11/october_2007_web_server_survey.html (accessed November 7th 2007)

7 “The Free Software Definition”, http://www.gnu.org/philosophy/free-sw.html (accessed November 7th 2007)

8 It should be noted that a firm is actually providing such a service on a large scale: Sourceforge.net (http://sourceforge.net), which hosts around 160,000 open source projects (as of November 2007). Most are barely working prototypes, but some of the most widely used software is partly or completely managed in this infrastructure. Their business model is based on advertisement, and more recently, being a market place between suppliers and consumers of services around open source software.

9 “MySQL Enterprise Production Support”, http://www.mysql.com/products/enterprise/support.html (accessed November 7th 2007)

10 There is also a specific value in using open source software, such as not being tied up to a given provider (a third party would have the right to maintain and modify the code) or the better quality usually associated with open processes, but they are not related to the value of the good we consider here (driving the extension) but to the simple availability of the software and its code.

11 An open source license doesn't mean that intellectual property rights are given up. The code can still belong to the firm even if it is freely available. A strong and exclusive brand can be developed without conflicting with open source constraints. Other (proprietary) licenses can also be offered in addition to the open source one, as long as one is owner of the code.

12 The original open source license is the GPL (GNU Public License) which is very restrictive, and forces software build or even shipped with it to be open source as well. Other licenses such as the Lesser GPL (LGPL), the Apache 2 or the BSD license allow software licensed with them to be combined with proprietary software. From a strict legal point of view, as long as you don't publish it, you can combine proprietary software with GPL software, but practically it scares off many people.

13 An annotated version which explains the rationale behind each section can be found here: http://www.opensource.org/docs/definition.php

14 See “Creative Commons Attribution 2.5 Generic”, http://creativecommons.org/licenses/by/2.5/ (accessed November 8th 2007)