Open Source Licensing in Mixed Markets, or Why Open Source Software Does Not Succeed

Gaudeul, Alexia
Department of Economics, University of East Anglia, ESRC Centre for Competition Policy

29. July 2008

Open Source Licensing in Mixed Markets, or Why Open Source Software Does Not Succeed

by

Alexia Gaudeul
Dept. of Economics and ESRC Centre for Competition Policy
University of East Anglia

CCP Working Paper 08-2

The support of the Economic and Social Research Council is gratefully acknowledged.

ISSN 1745-9648

Open Source Licensing in Mixed Markets, or Why Open Source Software Does Not Succeed*

Alexia Gaudeul†

July 29, 2008

Abstract

The rivalry between developers of open source and proprietary software encourages open source developers to court users and respond to their needs. If the open source developer wants to promote her own open source standard and solutions, she may choose liberal license terms such as those of the Berkeley Software Distribution as proprietary developers will then find it easier to adopt her standard in their products. If she wants to promote the use of open source software *per se*, she may use more restrictive license terms such as the General Public License to discourage proprietary appropriation of her effort. I show that open source software that comes late into a market will be less likely than more innovative open source software to be compatible with proprietary software, but is also more likely to be made more accessible to inexperienced users.

Keywords: Open Source; Software; Standards; Compatibility; Network Effects; Duopoly; Mixed Markets; Intellectual Property; Copyright; Licensing.

JEL Classifications: D23, H41, L13, L22, L31, L86, O34, O38

*Early drafts of this article were written while I was an EC-funded Marie Curie Research Fellow at the University of Southampton. Jacques Crémer at the University of Toulouse, Robin Mason at the University of Southampton and Bruce Lyons at the University of East Anglia provided helpful advice. Thang To at the ESRC CCP provided very able assistance with data collection. This paper was inspired by a case study of the (LA)TEX project (Gaudeul 2007). The second part of the title is inspired by Bonaccorsi and Rossi's 'Why Open Source Software Can Succeed' (2003). The paper was presented at the OSSEMP workshop in conjunction with the Third International Conference on Open Source Systems in Limerick in June 2007, at the EEA conference in Amsterdam in August 2005, at the Toulouse Workshop on Open Source Software and Intellectual Property in the Software Industry in January 2005, at the Open Source Software conference 'Autour du Libre' at ENST Bretagne in Brest in May 2004, and in seminars at the University of Strathclyde in Glasgow and at the INRA GAEL in Grenoble. Support from the ESRC and from the European Commission is gratefully acknowledged.

†School of Economics and ESRC Centre for Competition Policy, University of East Anglia, Norwich. email: a.gaudeul@uea.ac.uk, website: http://agaudeul.free.fr

1 Introduction

Why is open source software ('OSS') widely used in some markets and development areas and not others? Why is it so difficult in some cases to use OSS alongside proprietary software ('PS') because of incompatibility problems, while in some other cases both type of software are used on the same types of platforms and use the same standards and may even be integrated seamlessly into each other? Why do some open source developers ('OSD') decide to adopt liberal license terms that allow integration of the open source ('OS') standard and its associated implementation into proprietary software ('PS') while others seek to exclude proprietary use by adopting more restrictive license terms? Why do some proprietary software developers ('PSD') choose to develop add-ons and interfaces to OS products while other PSDs develop software independently? This paper considers those questions by analyzing competition between a proprietary developer who wishes to maximize profits from selling his product on the market, and an open source developer with different motivations:

Motivations: One set of motivations for the OSD is own use or enjoyment; she wants to develop software she needs or enjoys developing (von Hippel, 2005). Another set of motivations combine to make her want others to use her software. This may be because she benefits from network effects, direct (the more people use software, the higher is its utility, for example communication software), or indirect (users may convert into developers who will then improve the software, contribute their expertise and knowledge and provide peer review (von Krogh, Spaeth, and Lakhani, 2003)). It may also be that she derives prestige and reputation from the software's success (Lerner and Tirole, 2002).

Goals: In addition to those two sets of motivations, I will want to make the difference between whether the OSD focuses on pragmatic goals, typical of firms (Bonaccorsi and Rossi, 2006), such as getting her software's solutions and standards adopted, or on ideological goals, typical of individuals engaged in OS development, such as protecting and promoting open-source development methods and the open-source community (again, Bonaccorsi and Rossi, 2006).

- **Ideological goals** are those of a developer who cares about open source values such as code reciprocation and software freedom. She cares about providing software that is free (as in 'free beer'), freely modifiable and with specifications that are open. In this case she will not welcome what she would consider as 'hijacking' of her code by a proprietary developer.

She may then choose restrictive license terms such as the GPL, that make it difficult for proprietary developers to make use of her code or adopt her standard.

- **Pragmatic goals** are those of a developer who is interested in the technological, problem solving aspect of OSS. She may then choose liberal license terms, such as the BSD, that facilitate the integration of her solutions and standard into proprietary software and thus facilitate compatibility between her software and proprietary software.

I will examine in this paper how such different and potentially conflicting goals translate into market outcomes: market share, type of users served, license used, compatibility with the existing standard and development of an user interface.

Licensing: The open source developer will have the choice between the General Public License ('GPL')[1] and a license of the type of the Berkeley Software Distribution ('BSD').[2][3] The two licenses authorize anybody to use, distribute or modify the project's code for free, subject to acknowledging previous contributions and, in the case of the GPL, subject to distributing the modifications under the same GPL license. The BSD does not impose this later restriction: it allows developers to distribute modifications and improvements under other licenses, including under proprietary licenses. Because of its license, open source software is essentially free to use. Unlike open source licenses, proprietary licenses prohibit the unauthorized use, replication and modification of the product by others. The owner of the project can then sell the right to use his product.

The choice of license between the GPL and the BSD is the source of great controversy in the OS community. The open source software model, promoted by Bruce Perens, tries to encourage the involvement of commercial developers by encouraging the use of the BSD. The free software model, promoted by Richard Stallman, is more averse to involvement by commercial developers and promotes the GPL. That reluctance is informed by the bad experience of the hijacking of the development of Unix by AT&T, which was at the origin of the GNU project ('**G**nu is **N**ot **U**nix').

[1] http://www.gnu.org/copyleft/gpl.html

[2] http://www.opensource.org/licenses/bsd-license.php

[3] This does not imply of course that other types of open source licenses are not covered, only that they either belong to the same family of licenses as the GPL or the BSD, or that the way they differ from those two does not have a bearing on the analysis that is made in the model.

Literature: This paper is part of the literature on the coexistence of open-source and proprietary software projects in a competitive setting. OSS provides fringe competition that may discipline big monopolistic players such as Microsoft. It also provides an opportunity for software firms to collaborate in the development of OS industry-wide standards with no fear of seeing their efforts hijacked by others. OSS has made significant inroads in many areas of software development, from servers (Apache) and mail management systems (Sendmail) to operating systems (Linux), browsers (Mozilla) and typesetting engines (TEX). This leads to hopes (and fears) that OSS will come to replace proprietary software: 'for equal quality, consumers will prefer a free product to a paying one' (Schmidt and Schnitzer, 2003), 'OSS can achieve better quality and faster adaptation to technological change' (Kuan, 2002), 'the OS development method is more efficient than closed source development methods' (Johnson, 2002 and Johnson, 2006), 'OSS is open to innovation from many quarters' (von Hippel, 1994 and von Hippel, 1998), 'OSS is more flexible and offers better control of its internal working', etc...

Evidence shows that OSS breeds a new and more efficient 'private-collective' innovation model where OS and proprietary development methods support each other (von Krogh and von Hippel, 2003). Koenig, 2004 offers examples of such collaborative innovation in a list of for-profit OS strategies, followed by such companies as Oracle, IBM, HP or Red Hat. Mustonen, 2005 argues that PSDs will encourage and support OSS in order to promote their own standards as common standards, as done by Adobe for example. They may also do so to gain some control over the OS standard and influence its development, as done with Linux by IBM and with Java by Sun Microsystems. Case studies of the markets for operating systems, servers and web browsers do actually show that competition by OSS tends to accelerate the pace of innovation across the whole industry (Bitzer and Schröder, 2006).

Some differences persist however, and they relate to the ability of firms to provide better user interfaces, and also to attract users through subsidization of early users and advertising:

Interfaces: Bessen, 2006 argues that pre-packaged PS addresses common uses with limited feature sets while OSS targets users with more specialized and complex needs. Nichols and Twidale, 2003 point out how OSD usually have preferences in terms of user interface that differ from those of the common end-user. They tend to prefer command line based interfaces with many shortcuts, as those allow direct access to the basic functions of the software. They tend to dislike the more intuitive What You See Is What You Get ("WYSIWYG") interfaces that automate frequently used tasks but are less flexible. Against all this, Franke and von Hippel, 2003 argue from a survey of Apache users that developers and end-users do not actually differ much in terms of their objectives and needs for development.

Proprietary development will have an organizational advantage over OS development. Raymond, 2002 mentions the advantage for a 'big player with a lot of money' in doing 'systematic user interface end user testing' as well as 'setting up large-scale focus group testing with end users'. OSDs on the other hand will face problems in coordination, such as when TEX developers faced the prospect of leading radical mid-life changes to their software (Gaudeul, 2007). Open source development appears to be too unruly and undirected to provide the stability and support users need. PSDs would thus benefit from their ability to reliably direct the work of others in a centralized way according to a well defined and enforceable strategy defined from the point of view of the customer.

Strategy: Proprietary development will also have a strategic advantage over OS development. Casadesus-Masanell and Ghemawat, 2006 show that PSD can subsidize purchases by the first users in order to build a user base, and then exploit the latecomers. This, as well as advertising, is not affordable for OSDs as OSS generates only limited income streams. The case of Microsoft provides a range of other strategies to counter the emergence of OSS. FUD tactics (spreading Fear, Uncertainty, and Doubt) underline how OSS is supposedly 'unsupported'.[4] Prices may be lowered for vulnerable consumers such as public administrations who get preferential deals and consumers in less developed countries who get offered Windows XP Starter Edition, a lower cost lesser quality version of Windows XP. Faced with this array of strategies, OSS may receive the support of governments through public subsidies, mandated adoption or information campaigns (Comino and Manenti, 2005).

This paper completes the above literature by clarifying the impact of the differences between the OS and proprietary development models on the structure of the software markets. I show that the market outcome in the competition between OS and proprietary software is affected by: (i) whether the open source developer is a precursor or a follower, and whether open source developers want to promote adoption of their standard or of their software; (ii) whether network effects are important and whether the majority of consumers are professionals or non-specialists; and (iii) how costly user-interface development is and whether open source software's intrinsic quality is higher than that of proprietary software. An open source developer who is a latecomer to the market will be less likely than an early entrant to make her product compatible with that of the proprietary developer, but she is also more likely to orient her software towards the non-specialist (inexperienced) user. Depending on the factors outlined above, a manager may seek compatibility with open source software, borrow open source code, offer interfaces to open

[4]Microsoft's proposed response to the emergence of open source software as a competitor can be found at http://www.catb.org/~esr/halloween/index.html. Those are commonly called the 'Halloween documents'.

source development, or in other cases, be better off developing well away of open source developers and users. The OSD may choose the BSD license if she is a precursor and has pragmatic goals, which are typical of firms involved in OS development. She may also do so even if her goals are ideological, but in a more limited number of cases, *i.e.* if this increases the number of users of OSS compared to a situation where compatibility would not be achievable, that is, if allowing use of her standard in proprietary products softens competition. I provide empirical support for the assertions and findings of the paper by considering the market conditions in a variety of software development areas.

2 A model

Consider two developers. They are identical in every respect, except that one chooses to develop software under an open source license and the other chooses to develop software under a proprietary license. For clarity of exposition, I will refer to the open source developer as 'she' and to the proprietary developer as 'he'.[5]

Quality: Consumers can choose between an open source and a proprietary software of quality q_o and q_p respectively. The 'o' subscript denotes the open source product and the 'p' subscript denotes the proprietary product. If $q_p > q_o$ (respectively $q_o > q_p$) then proprietary (respectively open source) software is of higher quality than open source (respectively proprietary) software. Quality can be interpreted as the number of software functionality if as in the empirical section they can be ranked from basic to more sophisticated features. It may also represent code quality. Consumers all agree on the quality of each software.

Interfaces: There is a mass 1 of consumers who differ in their software expertise. Mass $M < 1$ of consumers ('inexperienced' consumers) need a WYSIWYG interface or a Windows port of the application or an extensive documentation for the program ('interfaces'). They cannot use software without those elements. The rest of the consumers do not need those elements. Significantly, I assume the OSD does not need those elements either, and faces cost c in providing them. I also assume the PD always develops interfaces, and considers this development as a sunk cost necessary to the marketing of his software. There are several ways to justify this assumption. The first is ease of exposition; adding the decision to develop interfaces or not for the PD makes

[5]This choice of convention does not necessarily suggest the likely gender of an open source developer. Only about 2-5% of OSDs are women (Hertel, Niedner, and Herrmann, 2003 or Ghosh, Glott, Krieger, and Robles, 2002), compared to about 25-28% of all developers in the proprietary software industry (trade publications). For some work on issues of gender in open source development, see Adam, 2004, Lin, 2006, Ratliff, 2005 or Lyman, 2005.

the solving of the model more involved. The second is practical; since a consumer cannot have access to the source code of proprietary software, it would be very difficult for them to port it to their own preferred platform, or to understand its functioning. It is therefore necessary for the PD to write documentation for the program and compile it for a variety of platforms to make it marketable. OSS on the other hand may still be used by experienced users even if no interfaces or documentation are provided. Indeed, the expert user will be able to make sense of the code or communicate with the developer directly.

Network effects: The consumers derive utility from the number of other users of the software they use. k will denote the strength of network effects. Usual sources of network effects – do consumers benefit from the use others make of the product? Do they exchange data using that product? Is that data exchange standardized? – are complemented by the impact that the continued use and development of the product will have on its quality over time.

Licensing: An open source developer can choose between the (liberal) BSD license that facilitates integration of her code and/or standard into a proprietary product, in which case compatibility can be achieved, and the (restrictive) GPL license that makes it more difficult and negates most benefits from doing so – the PSD cannot appropriate any improvement to the GPL standard. Conversely, the specifications of the PS may be made public so the OSD may achieve compatibility with PS.

Consumers: Suppose the two competing products, open source (o) and proprietary (p), of quality q_o and q_p respectively, are used by mass n_o and n_p of consumers respectively. The open source product is free, while the proprietary product is priced at p.

If the two products are **incompatible**, then a consumer who chooses the open source software derives benefit

$$q_o + kn_o \tag{1}$$

if the OSS has interfaces or if the OS product does not have interfaces and the consumer is experienced. She derives no value from the software if the OSS does not have interfaces and she is inexperienced.

A consumers who chooses the proprietary software derives benefit

$$q_p + kn_p - p \tag{2}$$

from the proprietary product, whether she is experienced or not.

If the two products are **compatible**, then n_o and n_p are replaced in the equations by $n_e = n_o + n_p$.

Developers: The open source developer derives utility from using her own software (Raymond, 2001) and may also develop interfaces to make her software usable by all if that brings about positive network externalities, such as prestige, increased usage, high network value, etc.

Her utility is

$$U_o = q_o + kn - c \tag{3}$$

if she develops interfaces and

$$U_o = q_o + kn \tag{4}$$

if she does not develop interfaces.

- c reflects the costs of developing interfaces. The OSD incurs those costs fully as she does not derive any direct benefit from developing interfaces since she is an experienced user.
- n is 1) the number of users of her software if her goals are ideological, or 2) the number of consumers who use software based on her standard if her goals are pragmatic (see p.2).

The proprietary developer aims to maximize profit. Denote p the price of his software and n_p the number of its users. His profit is $\Pi = pn_p$.

Timing: I will assume development is sequential; either the OSD or the PSD develops first. I will call the first mover a precursor and the second mover a follower. The second mover observes the licensing choice and development choice of the first mover and then decides whether to develop and what to develop. This dynamic element in the model reflects a pattern of OS and proprietary development. Rather than occurring at the same time along different lines, they alternate leadership, catching up and borrowing from each other. The (LA)TEX case study provides an example of such dynamics in the development of software in the typesetting industry (see Figure 1 in Gaudeul, 2007 and section 5 in this paper). This dynamic element in the model also reflects differences between software that is pathbreaking and introduces new ideas *vs.* that which imitates and borrows from other software. As discussed in section 5, there is some indication that OSS is very often an imitation of existing software.

Once both developers have developed their product, the proprietary developer chooses the price p of his product and consumers choose which product to use. The open-source product is free.

I will use the concept of fulfilled expectation equilibrium (Katz and Shapiro, 1985) to determine the equilibrium product adoption and profits or utility of the developers. A fulfilled expectation equilibrium is such that equilibrium adoption is what consumers' expected equilibrium adoption

to be. I will assume a proprietary developer can always tip the market in his favor. This may be through a combination of advertising and/or low introductory pricing, all strategies that are not available to the same extent to an OS developer (see Casadesus-Masanell and Ghemawat, 2006 for an analysis of such forward looking pricing strategies).

3 Monopoly market

In this part, I consider a single developer with software under either the OS or the proprietary license.

Open source software: In a monopoly situation, the OSD may choose not to develop interfaces and be used only by portion $1 - M$ of users, so her utility is $U_o = q_o + k(1 - M)$, or she can develop interfaces and be used by all users, so her utility is $U_o = q_o + k - c$. She will choose the later s.t. $k > \frac{c}{M}$. Intuitively, high M (many inexperienced users) encourages the OSD to serve all by developing interfaces. High k (high network effects) sustains this tendency as gaining additional consumers become more valuable.

Proprietary software: The monopoly PSD will sell to all consumers at price $p = q_p + k$ and make profit $\Pi = q_p + k$.

4 Mixed market duopoly

This part examines the development, compatibility and licensing decisions of the OS and proprietary developers when they compete with each other in a two stage game.

In a first part I will analyze the situation where the OSD develops first. In a second part I will look into the case where the PSD is the first mover. A first mover will have to decide on whether to allow compatibility as well as on the licensing of its code and on whether to develop interfaces.

The OSD leader's choice between the BSD and the GPL license terms is important. Indeed, the BSD allows integration of her standard into a proprietary product, so the PSD may then choose to use the OS standard and build on it. The OSD may derive benefits from this as this means her standard will be used by more people. However, that also means users may prefer the implementation of her standard in its proprietary form to its OS implementation, for example if PS has more users, or if PS is of higher quality, or if PS is the only one offering interfaces. The OSD may therefore choose the GPL to avoid this.

If the PSD moves first, the later coming OS standard will not be integrated into his software so the OS follower will be indifferent between the GPL or the BSD license. The PSD may make the specifications of the proprietary software available so as to encourage the production of compatible or add-on products. Rather than adopt the proprietary standard, the OSD may decide to develop her own standard because incompatibility between OS and PS allows her to gain a higher share of the market than compatibility.

4.1 The OSD is a precursor

If the OSD is a precursor, then the timing of decisions is as follows:

- Stage 1: The OSD chooses her software license, either BSD or GPL and chooses whether to develop interfaces or not.
- Stage 2: The PSD chooses whether to adopt the OS standard and be compatible with OSS, or not adopting the OS standard and not be compatible. Adopting the OS standard is possible only if OSS is under the BSD.
- Stage 3: The PSD chooses the price p of his software and consumers simultaneously choose which software to adopt. Those who choose PS must pay for its use, those who choose OSS do not pay anything.

At any one specific stage, all agents observe all decisions made by all other agents in the preceding stages. The game will be solved by backward induction. The different situation in stage 3 are outlined in appendix A.

4.1.1 Stage 2: the compatibility decision of the PD

There are three cases: either the OSD developed interfaces, in which case both software are in direct competition so incompatibility is preferred by the PSD. Or the OSD did not develop interfaces, in which case the outcome depends on the OSD's licensing choice. If the OSD chose the GPL, then compatibility is not achievable and this will encourage the PD to serve all consumers. If the OSD chose the BSD, then the PD may accommodate OSS and serve only inexperienced consumers while adopting the OS standard. This is summarized in the proposition below:

Proposition 1 *A) If the OSD developed interfaces and if $k > q_o - q_p$, then the PD prefers incompatibility and gains the whole market. If $k \leq q_o - q_p$ then the PS does not enter the market, on which the OSD will have a monopoly.*

B) If the OSD did not develop interfaces and chose the GPL, then compatibility is not achievable and the proprietary developer chooses to serve all consumers s.t. $k \geq \frac{q_o-(1-M)q_p}{1-M^2}$, and otherwise serves only inexperienced consumers.

C) If the OSD did not develop interfaces and chose the BSD, then the PSD chooses incompatibility and serves all consumers s.t. $k \geq \frac{q_o-(1-M)q_p}{1-M}$, and otherwise prefers compatibility and serves only inexperienced consumers.

Proof. The proofs of the above are given in the appendix B of this paper. ■

Figure 1 below illustrates the above statements. IC denotes incompatibility, C denotes compatibility, Inexp. denotes inexperienced consumers.

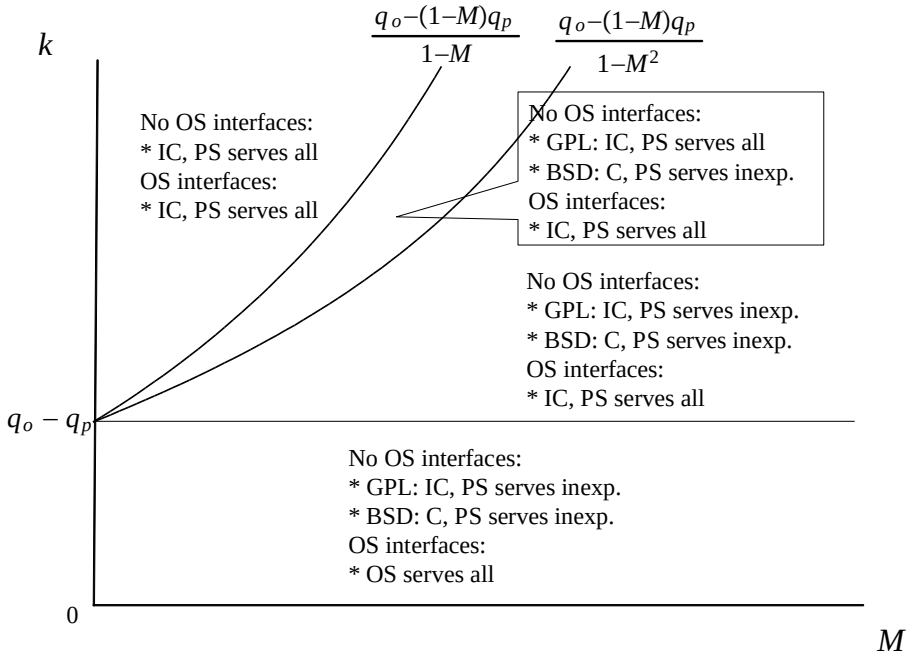

Figure 1: Market share and compatibility in a mixed market duopoly, as a function of OS interface and licensing choices.

When the OSD develops interfaces, the PD is in direct competition with OSS and prefers incompatibility whatever the choice of license by the OSD. The OSD will gain the whole market only

if its quality is higher than PS and network effects are low ($k \leq q_o - q_p$). Otherwise, the PD gains the whole market. This means that OSDs will be keen to develop interfaces only when network effects in their development area are low and the quality of their product is higher than that of competing proprietary software.

When the OSD does not develop interfaces and chooses the GPL, then the PD may decide not to serve the whole market and instead serve only inexperienced users (this happens if $k \leq \frac{q_o-(1-M)q_p}{1-M^2}$). Since the GPL prevents compatibility, this is with an incompatible product. When the OSD does not develop interfaces and chooses the BSD, the PD is more likely to serve only inexperienced users (this happens if $k \leq \frac{q_o-(1-M)q_p}{1-M}$) and when he does so, this is with a product that is compatible with OSS. This means that the OSD may prefer the BSD to the GPL not only in order to get her standard adopted but also in order to share the market when otherwise the PS would have monopolized it. Choosing the BSD expands the range where OSS or an OS standard is used by at least a portion of consumers compared to choosing the GPL. The BSD encourages compatibility which has two main advantages: for the developer with pragmatic goals, this gets her standard adopted by more users since it is integrated in a proprietary version. It also saves her the cost of developing interfaces. For the developer with ideological goals, offering compatibility may avoid head on competition whereby the PS would gain the whole market.

From this part, I can also point out that higher quality OSS does not necessarily dominate its market. The analysis in the article admits the possibility that OSS would be intrinsically better than PS ($q_p < q_o$) and yet PS stays on the market by developing interfaces for inexperienced users. This can happen as long as $q_p > q_o - k$. Note also that conversely, one may have $q_p \geq q_o$ and yet open source software maintains a share of the market. This happens if the PSD chooses to concentrate on inexperienced users. The case where $q_o > q_p$ and yet PS serves inexperienced users would validate the paradoxical perception by experienced OSS users that OSS is of higher quality than PS while at the same time inexperienced, PS users cannot fathom using OSS because it lacks the interfaces that are essential to them. The case where $q_o < q_p$ and yet OSS serves experienced users would validate the perception that users of OSS use OSS because PS is priced for other types of users that differ from users of OSS in their need for some specific, 'user-friendly' interfaces.

4.1.2 Stage 1: Interface development and licensing

Consider in this part whether the OSD will develop an interface for the end-user and if not, which license it will use. Conclusions in this part will depend on the OSD's goals, either ideological

or pragmatic. As seen previously, the OSD with pragmatic goals who chose the BSD may see a proprietary implementation of her standard developed by the PSD. There is therefore no point in her developing an interface if that is likely to happen. The OSD with ideological goals may develop an interface to her software if that allow her to gain market shares compared to not doing so. She will do so provided her software is of sufficiently better quality than that of the PSD and the gain in the number of users of her standard more than compensates for the cost c of developing the interface.

Proposition 2 *The OS precursor with pragmatic goals will never develop interfaces.*

She will strictly prefer the BSD to the GPL for any $k \leq \frac{q_o-(1-M)q_p}{1-M}$*, and is indifferent between the two licenses otherwise.*

The OS precursor with ideological goals will develop interfaces only if $q_o > q_p$ *and* $k \leq q_o - q_p$*, and if that is so, will do so s.t.* $k > \frac{c}{M}$*.*

She will strictly prefer the BSD only if $k \in [\frac{q_o-(1-M)q_p}{1-M^2}, \frac{q_o-(1-M)q_p}{1-M}]$*, and is indifferent between the two licenses otherwise.*

Proof. The proofs are given in the appendix C of this paper. ■

From this proposition, there is only a limited number of cases where OSS will develop interfaces. An OS developer with pragmatic goals derives no benefits from developing interfaces since the PSD will provide a proprietary implementation of her standard for use by inexperienced users. For interfaces to be developed, the OSD must have ideological goals, OS quality must be higher than that of equivalent proprietary software, and network effects must be neither too high (otherwise the PD would monopolize the market) not too low (otherwise, gaining experienced users would not be very valuable in terms of network effects). Non availability of an OS interface for OSS should therefore correlate with OSS being of lower quality than PS, or with the OSD having pragmatic goals. Availability of an OS interface for OSS should correlate with OSS being of higher quality than PS and should result with OSS gaining the whole market.

From this proposition, I can also conclude that the pragmatic OSD will almost always prefer the BSD (except when $k \geq \frac{q_o-(1-M)q_p}{1-M}$ when she is indifferent), while the ideological OSD will prefer the BSD only in a very limited range of case. This is because an OS developer with ideological goals does not care about those users who use the OS standard in its proprietary implementation. Since the OSD with pragmatic goals is more likely to choose the BSD and does not develop interfaces, there should be a link between choosing the BSD and not developing an interface.

Figure 2 below shows the utility of the OSD depending on her goals and on her interface and licensing decision. When the distinction between licensing choices or goals is not shown, this indicates that the utility is the same irrespective of licensing choices or goals.

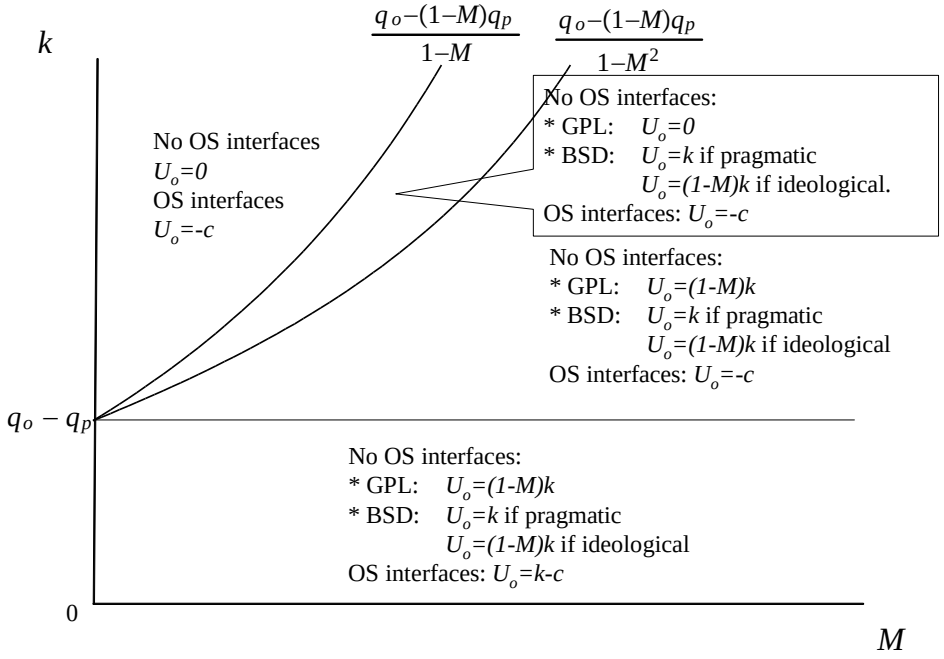

Figure 2: Utility for the OSD as a function of her goals and of her interface and licensing choices.

Consider now briefly the case where the OSD is a follower:

4.2 The OSD is a follower

If the OSD is a follower, then the timing of decisions is as follows:

- Stage 1: The PSD choose whether to make the specifications of his standard available or not.
- Stage 2: The OSD chooses her software license, either BSD or GPL and chooses whether to develop interfaces or not.
- Stage 3: The OSD chooses whether to adopt the proprietary standard and be compatible with PS, or not adopting the proprietary standard and not be compatible with PS. Adopting the proprietary standard is possible only if the PSD decided to allow this in stage 1.

- Stage 4: The PSD chooses the price of his software and consumers simultaneously choose which software to use. Those who choose PS must pay for its use, those who choose OSS do not pay anything.

At any one specific stage, all agents observe all decisions made by all other agents in the preceding stages. The game will be solved by backward induction. It will be necessary to distinguish between OSD with pragmatic and ideological goals in stage 3.

If the OS developer is the second mover ('OS follower'), then her standard will not be integrated in the proprietary software. That means that the incentive for the OS follower to develop interfaces will be the same as those of the precursor with ideological motivations, and this even if the OS follower has pragmatic goals. She is interested only in maximizing the number of users of her software, and this whether she has pragmatic or ideological goals. The OS follower is therefore indifferent between the GPL and the BSD. The PSD will prefer to open his standard only if that leads the OSD to adopt it, and she will do so only if that increases open source market share.

The above is summarized in the proposition below, that can be understood in comparison with proposition 2:

Proposition 3 *The OS follower with pragmatic goals will behave in the same way as the OS follower with ideological goals. She will develop interfaces whenever an OS leader with ideological goals would have done so. There will be compatibility with PS only when* $k \in [\frac{q_o-(1-M)q_p}{1-M^2}, \frac{q_o-(1-M)q_p}{1-M}]$.

Proof. Consider stage 4. This stage can be analyzed in exactly the same way as when the OSD is a leader (see appendix A). Consider now stage 3. If the PSD chose not to open his standard in stage 1, then no compatibility may occur, and the analysis is the same as when the OSD is a leader and chose the GPL. If the PSD chose to open his standard in stage 1, then the OSD will choose to adopt it only if that leads the PD to choose to serve only inexperienced users while not adopting the proprietary standard would lead the PD to monopolize the market. This happens only for $k \in [\frac{q_o-(1-M)q_p}{1-M^2}, \frac{q_o-(1-M)q_p}{1-M}]$. Consider now stage 2. The OSD will be indifferent between choosing the GPL or the BSD since her standard will not be adopted by the PD as she is a latecomer to the market. As before, she will develop interfaces only if $q_o > q_p$ and $k < q_o - q_p$, when this may gain her the whole market rather than sharing it with PS. Whether

her goals are ideological or pragmatic, she will develop the interface only s.t. $k - c > (1 - M)k$, that is, if $k \geq \frac{c}{M}$. ■

At this point, it is possible to compare OS precursors and followers and BSD and GPL software. The main question is whether precursors or followers will gain the highest market shares for OSS and for OS standards, and which license will be associated with the most successful OS software and/or standards.

BSD standards will generally be more successful than GPL standards because they will be adopted in proprietary software when GPL standard would not. However, software developed under the GPL software is more likely to offer interfaces and thus be easily available to all than BSD software. BSD software, when it coexists with PS that uses the OS standard, will serve professional or expert (experienced) users. Finally, the BSD license is more likely to be chosen by a precursor than by a follower, and by an OS developer with pragmatic goals than by an OS developer with ideological goals.

The standard will be OS in markets with lower network effects if OSS quality is higher than that of PS and the OSD develops an interface so all consumers use the OSS. The standard may also be OS when OSS is a precursor in its market as long as it is licensed under the BSD. In that case the OS standard is adopted by the PSD, the OS implementation of the OS standard serves the professional/specialist users and the rest use the proprietary implementation with its easy to use interfaces. This finding would explain the difference between professionals' markets, where interfaces for OSS are developed by proprietary firms, and non professionals' end-users' markets where the interface will be a key development area for the OS projects (as Gnome is for Linux for example).

5 Empirical support

I assumed in this paper that OS developers think strategically when choosing whether to devote efforts to developing interfaces for their product, or when choosing the license terms for their software. This assumption is supported by my case study of the history of (LA)TEX (Gaudeul, 2007). (LA)TEX powers various widely used typesetting systems. A long series of interviews with OS and PS developers who participated in the development of (LA)TEX revealed a series of interactions between OS and proprietary development. PSDs initially developed software to appeal to those users who were not able or not willing to use (LA)TEX in its OS implementation.

They were mindful of the need to identify such users, respond to their needs and differentiate from the OS offering. This is how for example proprietary implementations of (LA)TEX such as Scientific Workplace were able to gain large market shares. As the market for typesetting software expanded, some PSDs chose to develop typesetting systems independently of (LA)TEX (Quark, Framemaker...). Over time, they differentiated enough or provided output of sufficiently high quality to attract many of those users who previously used (LA)TEX. In later stages, interfaces were developed from within the OS community, with TEXLive and MiKTEX offering an easy to install (LA)TEX distributions and LYX offering a WYSIWYG interface for LATEX typesetting. This history thus evidences a variety of different patterns of cohabitation between OS and PS in one development area as the relative quality of competing software and degree of experience of consumers varied over time.

Comparison between BSD and GPL software also support this paper's theoretical predictions. I showed that OS precursors are more likely than followers to choose the BSD license and for their standards to be integrated in proprietary software. In practice, one indeed observes that BSD software such as (LA)TEX, Apache, Sendmail or Unix were all precursors and were all integrated into proprietary offering which gave them mass-market appeal. On the contrary, one observes that GPL software such as Linux, Gnome or Firefox (Mozilla) were all inspired by existing software (Unix, Windows and Netscape respectively) and were initially relegated to niche markets. Only with gradual improvements in their quality, notably in their interface, did they begin to make inroads into mass markets. One also notes that BSD software is essentially software for expert, professional users (Apache for example) while GPL software (such as Linux) is more likely to be developed as a hobby, to learn, to establish reputation or for ideological reason. This confirms the theoretical findings from the paper if one accepts that firms and professionals are more likely to have pragmatic goals, which motivate the choice of the BSD.

Empirical data on the software industry surveying open and proprietary applications over several product categories and operating systems is reported in appendix D, and underlines that OSS is generally of lower quality or with less functionality than PS and difficult to access for the majority of users as it is not available on Windows. This makes it significantly less popular than PS in most development areas.

1. OSS offered lower quality and less features than proprietary software. 50% of standard features were available in OSS on average, *vs.* 70% in proprietary software. However, in most categories, at least one OSS had as many features as the proprietary software with

the most features. This means that only few OSS have a chance to overtake PS. I assert in the paper that OSS that is available on Windows (has an interface) is likely to be of higher quality than corresponding proprietary software. I found indeed that OSS that is available on Windows is of significantly higher quality than OSS that is not available on Windows, and this in all categories and across all OS license terms. However, even OSS that is available on Windows is generally not of higher quality than proprietary software that is available on Windows, except for database software. This means there is only partial support for the assertion in the paper.

2. For software to be available in practice to the 95% of computer users who use the Windows operating system, it must be available under that platform. Being available on the Windows platform was used as an indicator for whether interfaces are available or not, since in the model, providing interfaces means making the software accessible to a wider audience, in the same way as making software available under Windows makes it available to most. Only 75% of open source software were found to be available on Windows, *vs.* 95% for proprietary software. However OSS was more likely than PS to be available on Mac, Unix, Linux. This confirms that OS developers often prefer not to devote efforts to developing interfaces. The analysis of the data also shows there is strong evidence that OSS that is available on Windows gets higher market shares than OSS that is not available on Windows, all of which supports the model's conclusions.

3. OSS rarely got more than 20% of the market according to my measures. In the instant messengers category, OSS could not compete against free proprietary software, each having its own proprietary protocol (AOL, Yahoo! and Microsoft). OSS was not very successful either in 'professionals' markets: In graphic design, incompatible products with proprietary standards dominate (QuarkXPress, Adobe InDesign, Microsoft Publisher, Framemaker, Apple Pages). The only significant graphic design package, Scribus, cannot read or write the native file formats of commercial programs. In database software, where the common standard is sql, proprietary firms dominate too (Microsoft Access, FileMaker Pro) and dual licensed OSS (MySQL) uses the common standard. The only area where OSS achieves success is the web browser category where network effects are relatively low, standards are open (html) and OSS offers good interfaces (Mozilla Firefox is an example). Those findings confirm the difficulty for OS software to gain a significant share of the market. Either it is relegated to a minority of specialist users and its standard is adopted in proprietary software that then serves the majority of inexperienced users, or it is displaced by

proprietary software with its own standard when specialist users represent a majority of the market.

4. Whether the GPL or the BSD was chosen depended on the development area. This confirms that the choice of license is at least partly dependent on the conditions in the market, as posited in the paper. An assertion from the paper is that BSD software is less likely than GPL software to offer interfaces, because BSD standards are likely to be taken up by proprietary developers, who will take care of the interface. Data shows however that all BSD software is available on Windows, while only 65% of GPL software is. This would seem to contradict the above. However, higher quality of BSD software may be what encourages the development of OS interfaces for BSD software (BSD software stands a higher chance to win in frontal competition with PS). It is difficult to compare the quality of GPL and BSD software, as it is only in the web browser category that both BSD and GPL are used. However, I find that BSD software is of significantly higher average quality than GPL software, and of equal average quality than proprietary software in that category. I also find that for database software, where all OSS is under the BSD, OSS is of higher average quality than proprietary software. There is therefore some evidence that BSD software is of higher quality than GPL software, which might explain its wider availability on Windows.

5. Data relating age and innovativeness of OS *vs.* proprietary software were not conclusive (see appendix D). Other sources of data must therefore be used: In a study by Klincewicz, 2005 of the 500 most popular open source projects on Sourceforge, the main OSS repository, about 87% of the projects were deemed non-innovative and about 10% were considered as existing technology that was adapted for use on a new development platform. One per cent only were considered as radical breakthrough and only 13% of the projects were not direct imitations of existing products. Whether that level of innovativeness is higher or lower than that of proprietary software is not discussed, but the perception of open source software as essentially ‘me too’ products seems to be widespread (The Economist, 2006). This would explain why the BSD license is rarely used in OS project (as seen in the paper, the BSD license is primarily used by OS precursors).

6 Conclusion

I offered in this model a typology of the licensing, development and orientation of competing OS and proprietary developers. The outcome will be affected by whether the OS developer is

a precursor or a follower and by whether the OS developer wants to promote adoption of her standard or of her software (pragmatic *vs.* ideological goals). The outcome is also affected by whether network effects are important and whether consumers are in their majority professionals (experienced) or non-specialists (inexperienced). The outcome finally depends on how costly user-interface development is and on whether OSS's intrinsic quality is higher than that of proprietary software. I draw several observations from the model. Those can be used as guidelines for the analysis of the development history and present competitive situation of OSS projects.

I consider the availability of user-oriented developments such as an interface that is easy to learn to use, a distribution that is easy to install, support for users, documentation and so on. I show that OS precursors will be less likely than OS followers to engage in such activities and such developments, especially if their goals are pragmatic. They will prefer to let PSDs take on such user support and enhancements. An OS precursor will engage in such activities only if her product is of higher quality than PS and development of interfaces gains it a monopoly on the market. Early and innovative OSS will be used mainly by developers or professionals, which does not however mean OSS will have a minority share of the market, as professionals and/or developers may form the majority share of some markets.

Innovative OS development, that is, development that is groundbreaking, anticipates the needs of the common user and predates proprietary development, may benefit from being under the BSD license. On the other hand, OS development that follows in the footsteps of proprietary development is not affected by its choice of license. Innovative OSS is more likely to be compatible with proprietary software than an OS follower is, as a proprietary follower is more motivated than an OS follower to make his software compatible with the leading software. This is all the more true when the OSD's development objective is to foster adoption of her standard and solutions (pragmatic goals) rather than to foster adoption of open source software *per se* (ideological goals). In so far as compatibility is socially desirable, as it increases the joint value of proprietary and OS software, and in so far as letting the proprietary follower adopt OS standard is also socially desirable, since a single standard is desirable, then an OS precursor who chooses the BSD will generate higher welfare than an OS follower or than an OS precursor who chooses the GPL.

Further work in this area should focus on empirical investigations of the link between 1) the market positioning of OS and the compatibility and licensing decisions of OSDs on the one hand, and 2) the chronology of innovation in software design and the strength and origin of network effects in OS development on the other hand. There is also work to do in comparing the development dynamics of OSS that is under the BSD license with that of OSS that is under the GPL license.

References

ADAM, A. E. (2004): "Hacking into Hacking: Gender and the Hacker Phenomenon," *ACM SIGCAS Computers and Society*, 32(7).

BESSEN, J. (2006): "The Economics of Open Source Software Development," in *Open Source Software: Free Provision Of Complex Public Goods*, ed. by J. Bitzer, and P. J. H. Schröder, pp. 57–81. Elsevier: Amsterdam, http://ssrn.com/abstract=588763.

BITZER, J., AND P. J. H. SCHRÖDER (2006): "The impact of entry and competition by open source software on innovation activity," in *The Economics of Open Source Software Development*, ed. by J. Bitzer, and P. J. H. Schröder, pp. 219–246. Elsevier: Amsterdam.

BONACCORSI, A., AND C. ROSSI (2003): "Why open source sofware can succeed," *Research Policy*, 32, 1243–1258.

——— (2006): "Comparing Motivations of Individual Programmers and Firms to Take Part in the Open Source Movement: From Community to Business," *Knowledge, Technology, and Policy*, 18(4), 40–64.

CASADESUS-MASANELL, R., AND P. GHEMAWAT (2006): "Dynamic mixed duopoly: A model motivated by Linux vs. Windows," *Management Science*, 52(7), 1072–1085.

COMINO, S., AND F. M. MANENTI (2005): "Government Policies Supporting Open Source Software for the Mass Market," *Review of Industrial Organization*, 26, 217–240.

FRANKE, N., AND E. VON HIPPEL (2003): "Satisfying heterogeneous user needs via innovation toolkits: the case of Apache security software," *Research Policy, Special Issue on Open Source Software*, 32(7), 1199–1215.

GAUDEUL, A. (2007): "Do Open Source Developers Respond to Competition?: The (LA)TEX case study.," *Review of Network Economics*, 6(2), 239–263.

GHOSH, R. A., R. GLOTT, B. KRIEGER, AND G. ROBLES (2002): "Free/Libre and Open Source Software: Survey and Study," Part IV: Survey of Developers.

HERTEL, G., S. NIEDNER, AND S. HERRMANN (2003): "Motivation of software developers in Open Source projects: an Internet-based survey of contributors to the Linux Kernel," *Research Policy*, 32(7), 1159–1177.

JOHNSON, J. P. (2002): "Open Source Software: Private Provision of a Public Good," *Journal of Economics & Management Strategy*, 11(4), 637–662.

JOHNSON, J. P. (2006): "Collaboration, Peer Review and Open Source Software," *Information Economics and Policy*, 18(4), 477–497.

KATZ, M., AND C. SHAPIRO (1985): "Network Externalities, Competition and Compatibility," *American Economic Review*, 75, 424–440.

KLINCEWICZ, K. (2005): "Innovativeness of open source software projects," Discussion paper, Tokyo Institute of Technology and Warsaw University.

KOENIG, J. (2004): "Seven open source business strategies for competitive advantage," IT Manager's Journal, http://www.itmanagersjournal.com/feature/314.

KUAN, J. (2002): "Open Source Software as Lead User's Make or Buy Decision: A study of Open and Closed Source Quality," Stanford Institute for Economic Policy Research, Stanford University.

LERNER, J., AND J. TIROLE (2002): "Some Simple Economics of Open Source," *Journal of Industrial Economics*, 50(2), 197–234.

LIN, Y. (2006): "Women in Free/Libre Open Source Software Development," in *The Encyclopedia of Gender and Information Technology*, ed. by E. M. Trauth, vol. 2, pp. 1286–1291. Idea Group Publishing: Hershey, PA.

LYMAN, J. (2005): "Getting in touch with the feminine side of open source," NewsForge, http://www.newsforge.com/articles/05/08/08/1449259.shtml.

MUSTONEN, M. (2005): "When Does a Firm Support Substitute Open Source Programming?," *Journal of Economics & Management Strategy*, 14(1), 121–139.

NICHOLS, D., AND M. TWIDALE (2003): "The usability of open source software," *First Monday*, 8(1).

RATLIFF, C. (2005): "Gender and Open Source," http://culturecat.net/node/889.

RAYMOND, E. (2001): *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary.* O'Reilly Media, Inc.

——— (2002): "Why Open Source will rule," ZDNet, March 28.

SCHMIDT, K., AND M. SCHNITZER (2003): "Public Subsidies for Open-Source? Some Economic Policy Issues of the Software Market," *Harvard Journal of Law and Technology*, 26, 473–505.

THE ECONOMIST (2006): "Open-source business," March 16th 2006.

VON HIPPEL, E. (1994): ""Sticky Information" and the Locus of Problem Solving: Implications for Innovation," *Management Science*, 40(4), 429–439.

——— (1998): "Economics of Product Development by Users: The Impact of "Sticky" Local Information," *Management Science*, 44(5), 629–644.

——— (2005): *Democratizing Innovation*. The MIT Press: Cambridge, MA.

VON KROGH, G., S. SPAETH, AND K. R. LAKHANI (2003): "Community, joining, and specialization in open source software innovation: a case study," *Research Policy*, 32, 1217–1241.

VON KROGH, G., AND E. VON HIPPEL (2003): "Special Issue on Open Source Software Development," *Research Policy*, 32, 1149–1157.

A Stage 3: the pricing and consumption decisions

Denote n_p^e, n_o^e the consumers' expected share of each software.

- **1**. Suppose the OSD **developed** interfaces in stage 2.
  - **1.a.** Suppose both software are **incompatible**. Consumers, whether inexperienced or not, will choose PS only if $q_o + kn_o^e < q_p + kn_p^e - p$. The PSD may price at $p = q_p - q_o + k(n_p^e - n_o^e)$ and if so is chosen by all and gains the whole market. In a fulfilled expectation equilibrium, one then has $n_e = 1$. The PSD then makes profit $\Pi = q_p - q_o + k$. This strategy is profitable only if $k > q_o - q_p$.
  - **1.b.** Suppose both software are **compatible**. All consumers use software, so $n_e = 1$. Consumers, whether inexperienced or not, will choose PS only if $q_o + k < q_p + k - p$. The PSD may price at $p = q_p - q_o$ and gain the whole market. The PSD then makes profit $\Pi = q_p - q_o$. This strategy is profitable only if $q_p > q_o$. If this is not the case then the PS does not enter the market.

- **2**. Suppose now that the OSD **did not develop** interfaces in stage 2.
  - **2.a**. Suppose both software are **incompatible**. Inexperienced consumers only have access to PS and buy it s.t.

$$q_p + kn_p^e - p \geq 0 \tag{5}$$

Experienced consumers buy the PS only s.t.

$$q_p + kn_p^e - p \geq q_o + kn_o^e \tag{6}$$

There are thus two cases,

* the PSD sets $p = q_p - q_o + k(n_p^e - n_o^e)$ and serves all consumers. In a Fulfilled Expectation Equilibrium, $n_p = 1$ and $n_o = 0$, so that $p = q_p - q_o + k$ and proprietary profit is $\Pi = q_p - q_o + k$. This strategy is profitable only if $k > q_o - q_p$.
* or the PSD sets $p = q_p + kn_p^e$ and serves only inexperienced consumers. In a FEE, $n_p = M$ and $n_o = 1 - M$ and the PSD makes profit $\Pi = M(q_p + kM)$. This is always profitable.

The proprietary developer will thus choose to serve all consumers s.t.

$$q_p - q_o + k \geq M(q_p + kM) \tag{7}$$

which can be rewritten

$$k \geq \frac{q_o - (1 - M)q_p}{1 - M^2} \tag{8}$$

  - **2.b**. Suppose both software are **compatible**. Then either the PSD sells to inexperienced consumers only with price $p = q_p + k$ and makes profit $\Pi = M(q_p + k)$ or the PSD sells to all at price $p = q_p - q_o$ and makes profit $\Pi = q_p - q_o$. He will choose to serve all consumers s.t.

$$q_p - q_o \geq M(q_p + k)$$

which can be rewritten

$$k \leq \frac{q_p(1 - M) - q_o}{M}$$

B Stage 2: The compatibility decision

This part makes reference to appendix A.

- Consider first the case where the OSD chose the BSD and the PD may thus choose to make his software compatible with OSS.
  - When the OSD develops interfaces, then the PSD will always prefer incompatibility to compatibility, as this allows it to keep consumer captive and make them pay for his proprietary network of users.
  - When the OSD does not develop interfaces, there are then two cases:
    * If $q_o-(1-M)q_p>0$: Then, if there is compatibility, the PD prefers to serve inexperienced consumers and makes profit of $\Pi = M(q_p+k)$. He will always prefer this to incompatibility and serving inexperienced consumers. He will prefer this to incompatibility and serving all consumers only if $M(q_p+k) > q_p-q_o+k$, that is, if $k < \frac{q_o-q_p(1-M)}{1-M}$.
    * If $q_o-(1-M)q_p<0$ (A) Then if there is incompatibility, the PD prefers to serve all consumers and make profit q_p-q_o+k. He prefers this to compatibility and serving all consumers, and prefers this to compatibility and serving inexperienced consumers only if $q_p-q_o+k > M(q_p+k)$, that is, if $k > \frac{q_o-q_p(1-M)}{1-M}$, that is, always since $q_o-(1-M)q_p<0$ from (A).
    * This can be summarized by saying that if the OSD does not develop interfaces, the PD prefers incompatibility and serves all consumers for any $k \geq \frac{q_o-q_p(1-M)}{1-M}$, and prefers compatibility and serves inexperienced consumers for any $k \leq \frac{q_o-q_p(1-M)}{1-M}$.

- Consider now the case where the OSD chose the GPL. Then the PD may not make his software compatible with OSS. This does not change the situation when the OSD develops interfaces, since in that case the PD did prefer no compatibility. If the OSD does not develop interfaces, then the PD will serve all consumers for any $k \geq \frac{q_o-(1-M)q_p}{1-M^2}$, and will serve only inexperienced consumers for $k \leq \frac{q_o-(1-M)q_p}{1-M^2}$.

C Stage 1: OS interfaces and licensing

- Consider first the case where the OSD developed interfaces. If $k \geq q_o-q_p$ she gets no share of the market and PS is incompatible with OSS, so her utility is $U_o=-c$. Therefore, the OSD never develops interfaces for any $k \geq q_o-q_p$. If $k \leq q_o-q_p$ she gains the whole market and her utility is $U_o=k-c$. Suppose she does not develop interfaces and

$k \leq q_o - q_p$. Then, if she chose the GPL, there will be incompatibility and she will gain only experienced users, so her utility is $U_o = (1 - M)k$. If she chose the BSD, then there will be compatibility, so the OSD with pragmatic goals gets her standard adopted by all and gains utility $U_o = k$, while the OSD with ideological goals does not care about adoption of her standard by the PD and gets utility $U_o = (1 - M)k$. Therefore, for $k \leq q_o - q_p$, the OSD with pragmatic goals will choose the BSD and gain utility $U_o = k$, while the OSD with ideological goals will be indifferent between the BSD and the GPL and will develop interfaces s.t. $k - c \geq (1 - M)k$, that is, if $k \geq \frac{c}{M}$.

- Suppose now $k \geq q_o - q_p$. As we saw previously, this means the OSD will not develop interfaces. If $k \geq \frac{q_o-(1-M)q_p}{1-M}$, the PD will never choose compatibility with OSS and will gain the whole market, so $U_o = 0$ whatever the choice of license. If $k \in [\frac{q_o-(1-M)q_p}{1-M^2}, \frac{q_o-(1-M)q_p}{1-M}]$, then the PD will gain the whole market if compatibility is not possible, so the OSD will not choose the GPL in that domain and will prefer the BSD. Her utility will then be $U_o = k$ if her goals are pragmatic (her standard is adopted by all), and $U_o = (1 - M)k$ if her goals are ideological (her software is used only by experienced users). If $k \leq \frac{q_o-(1-M)q_p}{1-M^2}$, then the PD will serve only experienced users whether compatibility is possible or not, so the OSD with ideological goals is indifferent between the BSD and the GPL. The OSD with pragmatic goals however will prefer the BSD as the PD will then adopt her standard and her utility will be $U_o = k$ rather than $U_o = (1 - M)k$ if she had chosen the GPL.

D Empirical study

In order to motivate and illustrate the model, data was collected in May 2006 on the prices, supported platforms, number of key features, age, and popularity of software, licensed either under Proprietary, GPL, BSD or Dual licenses, across six categories (Word processors, back-up software, database software, graphical applications, instant messengers and web browser). 84 software were surveyed in total, about half of them proprietary, half of them open source, with 13 to 15 software in each category. The sample was drawn from four main sources: Amazon (mainly proprietary software), Download.com (mainly freeware and shareware), Sourceforge (mainly open source software) and Google Directory (any type of software). Those sources spanned the four main software marketing/distribution/development categories (shareware, freeware, open source and shelfware). The sample included software for non-specialists (word processors, instant messengers, web browser) and software oriented towards specialists and professionals (data-

base, graphic design).[6] The sample also includes software with higher network effects (instant messengers, word processors) and those with lower network effects (back up, graphic design). The sample finally included highly standardized categories and others with many different standards: there was little compatibility between systems for instant messaging, between graphic design packages or between back-up systems. There was more compatibility, with at least some basic standard everyone can use, in word processors (rtf), database (sql) and web browsers (html).

D.1 Popularity

The sampling was designed to identify the main OS and proprietary software in each development category. This resulted in an about equal number of OS and proprietary software being studied in each development category (Figure D1). Quite striking is that the BSD and dual licenses were never used in word processors, backup software, graphic design and instant messaging, while the GPL was never used in database development. The only area where BSD, dual and GPL licensed software cohabited was web browser development. There thus seems to be preferences for different OS licenses depending on the development area. Among proprietary software, which were divided between freeware and commercial software, freeware was very present in Internet communication software such as web browsers and instant messengers.

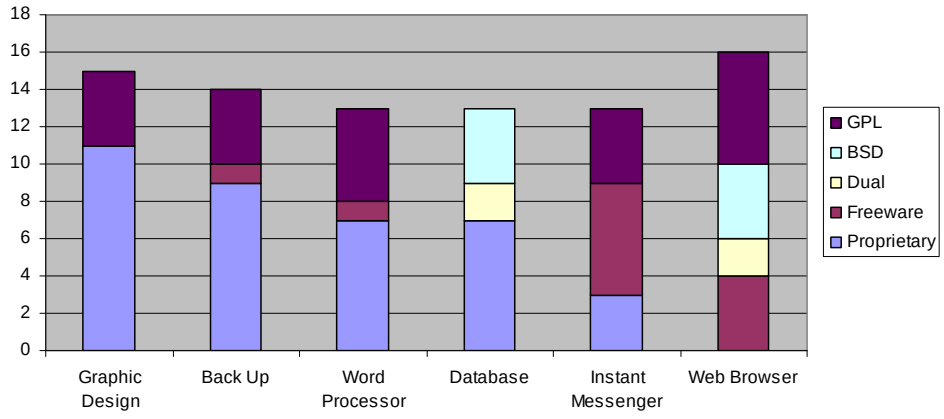

Figure D1: Number of software in the sample, by license and by category.

Popularity was measured according to three Web metrics: the number of links to the publisher's website (backlinks), Google's page rank for the website (from 0 to 10, higher is better), and

[6] In each category, some less sophisticated software did serve the non-specialist and others served more sophisticated users, which means for example that back up software did not fit neatly into either category.

the number of mentions of the software's name on the web. Those numbers, all collected from Google, the main search engine, indicate how easy it is to find the software on the Internet (Google Rank), but also how widely diffused the software is (number of mentions) and how often it is endorsed by others (backlinks). Those measures were strongly correlated with each other. Software popularity within its category was computed by averaging the proportion of the exponential of page rank, backlinks and mentions of that software within its category. Figure D2 below shows popularity of open source, dual license, proprietary software and freeware in their respective market categories.

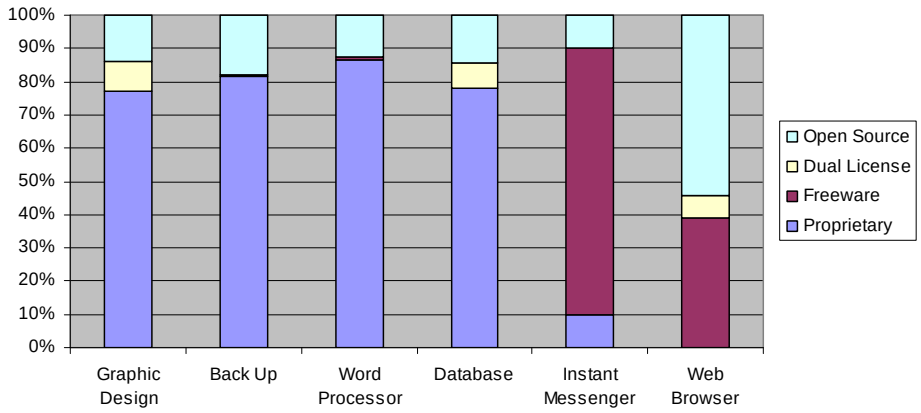

Figure D2: Popularity of software in the sample, by type and by category.

D.2 Quality

A list of the main features that could be available for software in a given category was established and was used to assess the quality of the software. Software with the most features was found to also include the features of software with less features. The number of features thus appeared to be an ordinal measure of quality. At the least, it could be considered as a measure of the capabilities of the software, without judging of the number of bugs, of the quality of output, or of the software's reliability and ease of use. Table D1 below summarizes the findings:

	Graphic Design			*Back Up*			*Word Processor*		
	Max	Mean	N	Max	Mean	M	Max	Mean	N
Proprietary	100%	71%	11	80%	67%	9	100%	86%	7
Freeware	.	.	0	40%	40%	1	50%	50%	1
Open Source	100%	43%	4	80%	60%	4	100%	38%	5
	Database			*Instant Messenger*			*Web Browser*		
	Max	Mean	N	Max	Mean	M	Max	Mean	N
Proprietary	100%	63%	7	71%	47%	3	.	.	0
Freeware	.	.	0	100%	67%	6	90%	70%	4
Open Source	100%	79%	4	43%	22%	4	100%	52%	10

Table D1: Percentage of key features provided, mean and maximum by license and by category (dual licenses not shown).

On this quality scale, OSS proved to be of significantly lower quality than proprietary software in word processors, instant messenger and graphic design. It was of lower quality in web browsers, of equal quality in backup software and of better quality in database software. On average, OSS included 50% of the maximum number of features, while proprietary software and freeware included 70%. In most categories however, at least one OSS and one PS had all the features on our list of main features.

There was little correlation between price and number of features, either on the whole or category by category, and popularity did not exhibit a link with price or our measure of quality. I cannot therefore go much further than comparing the popularity and quality of commercial software, freeware and OSS as was done above. This is especially so one could argue that less featured OSS may be faster and more reliable than proprietary software (an argument often made by OS proponents).

D.3 End-user orientation

Open source software was less likely than freeware and proprietary software to be available on Windows and PDA/Smartphones, but far more likely to be available on Mac, Unix and Linux. Indeed, only 75% of OSS was available on Windows, compared to about 95% of proprietary software while proprietary software was significantly less likely to be available on Unix and Linux (about 10% *vs.* about 65% for OSS) and also less likely to be available on Mac (about 40% *vs.* 60% for OSS). Overall therefore, OSS was less likely to be easily available to the mass market of Windows users but more likely to be available on many platforms than proprietary

software was. BSD software itself was more likely than GPL software to be available on the Windows platform.

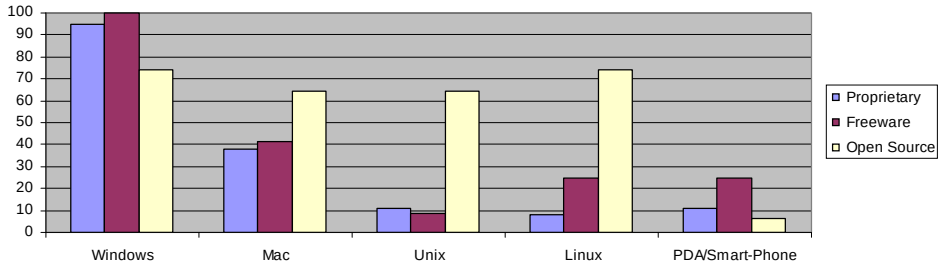

Figure D3: Platform availability, by category and by license terms.

D.4 Leadership

The date of inception of each software project under study was gathered in an attempt to determine which of OSS or proprietary software was the precursor in each different market. However, differences between the age of OS and proprietary software projects were not found to be statistically significant. Proprietary software tended to be either rather old (more than 10 years, maybe established software having gone through many versions) or very new (less than one year, maybe newly launched software by established software companies). OSS tended to be more evenly distributed in age, around 5 years in existence. This is maybe because very new OSS projects do not achieve sufficient popularity as fast as proprietary software with marketing tools can, and maybe also because the concept of OSS is too new for very old projects to exist. I can also posit that PS is able to persists for longer thanks to constant reinvestment in the development of the product and into related software. OSS on the other hand takes longer to establish and does not persist as an organization when the software becomes outdated.

Software that was old may be seen as either a precursor (in its own time) or as outdated (now), which makes age an improper measure of innovativeness. Because age and innovativeness cannot thus be related in a straightforward way, it is difficult to determine who were the leaders and who were the followers in the market. The version number or number of versions released was also tried as an indicator, but there are wide differences in the release cycles of the two types of software. For example, proprietary software may be repackaged with a new name rather than being given a new version number, while OSS may go through frequent minor releases that do not necessarily represent meaningful improvements.