Public provision of a private good: What is the point of the BSD license?*

Alex Gaudeul†

July 25, 2005

Abstract

Software is a potentially excludable public good. It is possible, at some cost, to exclude non-paying users from its consumption by using copyright law or technological restraints. Licensing the software under proprietary license terms makes of it a private good, licensing it under the BSD does not change the economic nature of the software while licensing it under the GPL artificially makes of it a pure public good. A project leader will prefer one or the other of those license terms depending on her software project's market potential and on the cost of developing it. The optimal licensing for a sequence of cumulative innovations and the impact of possible competition between rival software development teams are considered.

JEL Classifications: D23, D45, D71, D86, H41, H42, K11, L31, L86, O31, O32, O34.

Keywords: Open Source Software; Public Goods; Information Goods; Non-Profit; Volunteer Organisation; Intellectual Property; Copyright; Licensing; Innovation.

*I would like to thank Justin Johnson for remarks on a preliminary version, Sudipto Bhattacharya for references, participants at the Industrial Organization and Innovation Conference in Grenoble (2005) for their remarks and Daniel Zizzo for fruitful discussion.

†a.gaudeul@uea.ac.uk, University of East Anglia - Norwich and ESRC Centre for Competition Policy

Open-source development methods have attracted academic interest in recent years because of their apparently counter-intuitive economic characteristics and because of their spread beyond software development to many other areas. Online communities and other group communication mechanisms often function on open source principles. Independent news websites (blogs) have emerged as an alternative to established news media. Online databases that can be openly edited by anybody (wikis) generate alternative repositories of knowledge. Genetic research, among other research areas, has benefited from the use of open source methods. This paper will concentrate on the specific example of open-source software development but holds lessons for the organization of any of those various open-source community efforts.

Definitions Software development is generally protected by copyright. Copyright is the right to prevent others from using, modifying or selling your intellectual production without your permission. A copyright holder can then grant users a license underlining the rules under which they will be allowed to use, reproduce, distribute or modify the software.

Software can be licensed under three broad classes of licenses. It may have a proprietary license, be distributed under the rules of the Berkeley Software Distribution ("BSD"), or have a General Public License ("GPL") attached. The software may also be public domain and is then not protected by copyright. `www.opensource.org` provides a listing of other licenses but for the purpose of this paper, all licenses can be tallied up to one of those three groups of licenses. They are defined below:

If the original code was released under proprietary license terms, a modification to it will be under the copyright of the original author of the software. The author of the modification cannot claim copyright on the modified program. If the original code was under BSD license terms, a person who contributes code will have the copyright on her contribution and she can release that contribution under proprietary license terms.

However, the original code remains under the copyright of the original licensor. If the original code was released under GPL license terms, a person who contributes code will have the copyright on her contribution but she must release that contribution under the GPL. In other terms, she must copyleft her copyright. Finally, any work that is into the public domain cannot be copyrighted, although copyright on a modified version can be claimed. Choosing a BSD license is not equivalent to releasing the source code into the public domain because under the BSD the project leader keeps the copyright on the part of the software she developed herself.

Motivation This paper is partly motivated by a long-standing controversy in the open-source ('OS') community between supporters of the Berkeley Software Distribution ('BSD') 'open-source' license and supporters of General Public License ('GPL') 'free software'. On the one hand stand Bruce Perens and Eric Raymond of the Open Source Initiative (`www.opensource.org`), on the other Richard Stallman and participants in the development of the GNU/Linux operating system (`www.fsf.org`). GPL proponents want the code to be free; the source code of a software must always be available to all. BSD proponents want developers to be free; developers must be able to do what they want with the code, including making it proprietary, and thus not free in the GPL sense. GPL proponents think the BSD threatens the spirit of collaboration between free source developers because it allows the integration of BSD code into commercial, proprietary software. BSD supporters think the GPL unduly limits the domain of application for open-source software by discouraging participation and use by commercial software companies. GPL proponents argue that using the GPL helps discouraging the fragmentation of development teams into competing options ('forking'). The BSD on the other hand is alleged to lead to dissensions and distrust between developers as everyone fears somebody else will exploit the collective work in a proprietary version of the software. Faced with such potential lawsuits, BSD developers may then

prefer developing individually and not sharing their code rather than contributing to a public good that may then be appropriated by others. BSD supporters reply that competition between developers trying to produce commercially viable software induces a dynamic development process that leads to the development of software that is better suited to users' needs than GPL software generally is.

Modeling This paper takes the point of view of a developer with a software project. She will be called a project leader. She has the choice between three types of licenses, the BSD, the GPL and the proprietary license and she can draw on a pool of developers to develop her project. The project leader's licensing decision will determine whether developers will want to participate in the project and for what motives. This will influence the dynamics of the software development – how many developers will join the development team, whether there will there be competing development teams ('forking') and whether developers will release their work for free or instead make it proprietary ('hijacking'). The choice of the project leader will be determined by the expected size of the market for the software, the cost of developing the software, the cost of making the software proprietary, marketing it and protecting it from unauthorised copy and finally by the intrinsic value of the software project.

An example of the dilemma facing the project leader in its choice of license terms is the case of Netscape which in 1998 released its web browser under an open-source license. The Mozilla Organization (`www.mozilla.org`) was set up to coordinate open-source work on the project. Netscape initially imposed the Netscape Public License on the code, which would have allowed Netscape, and Netscape only, to use parts of Mozilla, the open source version of Netscape, in proprietary versions of the software. Faced with the unwillingness of developers to contribute under those conditions, Netscape had to change its license to a variant of the GPL, the Mozilla Public License.

Literature This paper is related to three different strands in the economic literature: papers exploring systems for the private provision of discrete public goods, papers trying to establish the optimal licensing schemes for sequences of innovations, and case studies examining the functioning of open-source software projects.

Private provision of public goods The title of this paper is a wordplay on an article by Justin Pappas Johnson, "Open Source Software: Private Provision of a Public Good" (Johnson (2002)). That article focused on the GPL license terms. The present paper points out that the GPL is only one specific open source software license, designed so as to minimize free riding, forking and hijacking in the development of a public good. This paper thus focuses on the other major open source license, the BSD. This licensing scheme may lead to the public development of a private good: a BSD licensed software may end up being made proprietary, even if it was the product of joint public development effort. One then has to wonder why a project leader would choose such a licensing scheme rather than using copyright law to impose either the GPL or a proprietary license on her project. The study of the BSD license terms proves more complicated than that of the GPL and this paper thus simplifies Johnson (2002) by assuming the cost and the value of development is the same for every developers. However, from that simplification, the paper introduces both the forking and development hijacking problem by assuming innovations take place over many periods in additive increments. The paper also introduces competition between software projects.

The model is also inspired by Varian (2004), from which it borrows some modeling techniques and a simplified version of a specific production rule, the one where the value of the public good is determined by the individual who contributed the "best effort" into its production. That production rule is particularly well fit to software production; there is usually one best way to develop a feature and there is no point including two ways to implement the same function into the same software.

Bergstrom, Blume, and Varian (1986) discusses private provision of public goods and, closer to this paper, Palfrey and Rosenthal (1984) discusses the private provision of discrete public goods. The present paper is not however a simple application of models of public good production to open source software; this would amount to comparing proprietary and public domain software. This paper is a study of the relatively recent original systems of public good production that operate under two main type of licenses, the GPL license terms and the BSD license terms. Choosing an open-source license leaves an innovator open to expropriation but broadens the pool of expertise she can draw from. This is a dilemma that is analyzed in Hellmann and Perotti (2004): the innovator must choose between keeping her idea in a firm or openly circulating it and sharing it with others.

Optimal licensing schemes This paper is not the first to underline the organizational consequences of the choice of an open source license. Other papers do exist that relate the choice of a specific open-source license term to the characteristics and dynamics of the development of that software. von Hippel and von Krogh (2003), for example, underlines the organizational consequences of the mix of private and collective motivations in open source innovation. Lerner and Tirole (2005b) is a preliminary, survey based study of the determinants of open source license choices. The article shows that projects that are oriented toward the needs of developers will more often be licensed under non-restrictive license terms (BSD) than projects oriented toward the needs of consumers who have limited programming abilities. Projects that could be profitably exploited on the market are thus generally licensed under the GPL. Projects that are oriented toward developers on the other hand do not need such protection because the benefit of making the software proprietary is limited: developers will find it easy to use the open-source version instead of the proprietary version, and there is only a limited prize to making the software proprietary as the market for it is limited.

Lerner and Tirole (2005b) also finds that projects with unrestricted license terms attract more contributors. Fershtman and Gandal (2005), which like Lerner and Tirole (2005b) is based on a survey of the Sourceforge open-source projects database (`www.sourceforge.net`), finds that projects with restrictive license terms will attract lower value individual contributions than projects with liberal license terms. The authors explain that contributors to a GPL project value the signaling aspect of a contribution to the project (being included in the list of contributors brings about prestige and better career opportunities for a developer). Contributors will thus contribute only up to what is needed to be listed among the development team. Contributors to a BSD project on the other hand will have another set of motivations that is not available to GPL developers: the prospect of making the software proprietary and exploiting proprietary versions derived from it. They will thus contribute up to the point where the software fits their needs.

Lerner and Tirole (2005a) offers a review of the actors' strategies in open source. They review papers such as Gaudeul (2005a) which compares the performances of volunteer and for profit organization in software production. Mustonen (2005) is another work that studies the interaction and co-existence of various license schemes in a software application domain.

The functioning of open-source software projects The present theoretical study of license terms choice is directly inspired by the (LA)TEX case study (Gaudeul (2003a), Gaudeul (2003b), Gaudeul (2005b)), as well as by the observation of the differing organizational dynamics of several open-source development projects. Unlike most projects that were previously analyzed in the open-source literature, TEX (`www.tug.org`) adopted a BSD-type license (The (LA)TEX Project Public License). Its license is therefore similar to Apache, a web server, and to BSD-Unix, an operating system. Proprietary versions of TEX were thus released over time and came in direct competi-

tion with the original open-source project. The dynamics of the development of that software thus differed in important ways from the development of the GPL licensed Linux operating system (`www.linux.org`). Another example of a BSD software whose logic of development was very different from that of Linux is BSD-Unix. There was fragmentation in the development of BSD-Unix, with various versions dealing with the specialized needs of a fraction of the original users of the software (FreeBSD (`www.freebsd.org`) for networking, OpenBSD (`www.openbsd.org`) for security, NetBSD (`www.netbsd.org`) for portability). Forking was more limited in the Linux project. BSD Unix was almost killed off by the threat of proprietary reappropriation of its code by AT&T. Some of the developers of Unix worked for AT&T's Bell Labs. This led USL, the assignee of AT&T's rights to Unix, to assert in 1993 that some parts of the various BSD versions of Unix were its property. A successor to USL, SCO (`www.sco.com`), renewed that type of claim in 2003 by asserting it owned the rights on all Unix software designed for Intel processors. The ensuing legal battles and uncertainties gave much of the impetus behind the development of the better protected GPL licensed Linux operating system. `www.levenez.com/unix` represents the development and forking history of Unix based systems, and Gaudeul (2005b) presents a similar time line of the development of the (LA)TEX typesetting system.

Organization of the paper In the first part, one project leader wants to develop one innovation, chooses her license and then chooses how many developers to include in her development team. Parameters that determine the choice of the developer between the proprietary, BSD and GPL license terms, are defined. In the second part, several innovations must be realized successively over time. Equivalently, there is a development programme which mandates that features be developed on top of each other. More simply, development may not occur in one go and developers build the software based on what was done by others before them; the development is the product of a

joint effort spread over time. The dynamics of development of the BSD and the GPL are compared. The third part examines the competitive setting; there is more than one project leader and the project leaders compete to attract developers to their team. The conclusion summarizes and discusses the main results of the paper.

1 The model

An individual (the project leader or 'PL') owns the copyright on a software project. The project consists of a sequence of innovations (features) $f \in \{1, ..., F\}$. Features must be developed in strict succession, one after the other (cumulative innovations), in the order of their ranking in the set $\{1, ..., F\}$. F is a function of how close to completion the software is when the PL asks for assistance, and is also a function of the complexity of the software. Each feature has the same value v to the M consumers, to the developers and to the project leader. v represents the usage value of the software. M represents the level of interest in the software in the wider, non-developer community. Mv is thus the market value of the software. There is a pool of developers who can develop the software at cost c. c is the opportunity cost to developing the software. c' is the cost to make a feature proprietary, it is the cost of setting up a proprietary organization that will protect the software from unauthorized copying and that will market the software.

$$v - c \tag{1}$$

is thus the net private value of the software to a developer, while

$$Mv - c' \tag{2}$$

is its net market value.

Assumption 1 $Mv - c' \geq 0$

This paper focuses on the case where the software to be developed is commercially viable. If $Mv - c' < 0$ then the software is never made proprietary because the investment c' in making it so is more than the gain Mv from selling the product on the market.

The game proceeds in several stages and is repeated as long as the project is not developed. Suppose feature $f \in \{1, ..., F\}$ was developed last period, and $f < F$. In this period, the PL wants to get feature $f + 1$ developed. The game is played according to the following sequence:

1. The project leader decides on the license terms to be applied to the code that embodies the sequence of innovation $\{1, ..., f\}$. The license can be BSD, GPL or proprietary.

2. The project leader distributes the source code to n developers under the license chosen in stage 1. Developers are indexed by $i \in [1, n] = N$.

3. Developer i chooses $a_i = (0, 1)$, with $a_i = 1$ if i develops the feature and $a_i = 0$ else. Developers do not observe the choice of other agents in the set N, but they know they all have the same cost of effort c, value for the feature v, as well as cost c' to make the feature proprietary. They also all know M the size of the market. Denote s_i the (possibly mixed) strategy of developer i. The vector $A = [a_1, a_2, ..., a_n]$ is realized.

4. The probability the feature is developed is $P(F(A))$ with $F(A) = \max_{a_i}[a_1, a_2, ..., a_n]$ and $P(1) = 1$ while $P(0) = 0$. This means that even if there is only one developer i such that $a_i = 1$, the feature is developed with probability 1. There are no synergies between developers.

5. Define $N' = \{i \in N \ / \ a_i = 1\}$. Developers in the subset $N' \subset N$ observe other members of N' and decide whether to incur cost c' to make their feature proprietary. Denote $c'_i = (0, 1)$ developer i's choice, with 1 denoting the choice to incur expense c'. This determines a subset $N'' \subset N' = \{i \in N' \ / \ c'_i = 1\}$.

6. Developers in N'' set prices $p_i, i \in N''$ for their feature.

7. Consumers observe the vector of prices $P = (p_i)_{i \in N''}$ and choose from whom to buy the feature.

8. If feature $f + 1$ was not developed in this period, then the game moves to the next period until the feature is developed. The per-period discount factor is δ.

9. If feature $f + 1$ was developed this period, then the game moves to the next feature, $f + 2$ to be developed, or if $f + 1 = F$, the game finishes. The discount factor from feature to feature is ρ.

δ is a function of the dynamism and efficiency of the open-source community and of the ease of contacting developers there. δ will be assumed to be zero in most of the paper. This can be interpreted as meaning that if developers that were first contacted chose not to develop the software, none later will. This can also mean that the idea for development is revealed only once to one or many developers, and will not circulate afterwards through a sequence of developers until developed.

ρ is a function of how easy it is to integrate, publish and circulate new features in the software. It is necessary for this to happen before development can move to the next feature.

The project leader can be interpreted as being an idea generator or a person with the charisma to direct the development of the software. She has got the blueprint for the software to be developed and she decides on the sequence of features/ideas to be developed. c is the investment to be put into each idea for their value to be realized.

Expending c gives one the copyright on the idea. The project leader can choose to incur that investment c herself, in which case she owns the idea and can sell it to others. However, it costs her c' to protect her copyright (proprietary license terms). She can also release the idea to a chosen set of developers and ask that they release its implementation for free (GPL license terms). Finally, the project leader can let any developer who implemented the idea exploit it freely. That developer is then free to sell the completed product/idea/feature to others, provided it expends cost c' to protect its copyright (BSD license terms).

Note that the project developer can limit the team size. She can filter whom to accept contributions from. This is a reasonable assumption even though it goes counter to the myth of open source communities being open to all contributions. OS development teams can in fact choose whether to be closed or open, that decision not necessarily being the decision of one person – it can be the result of a collective decision. This is the case for example of the Apache webserver development team, where developers have to go through several stages to become able to modify the Apache code more and more freely. The functioning of the Apache Software Foundation is explained in for example Hann, Roberts, Slaughter, and Fielding (2004). While a project leader will first wonder how to get developers to work on her project, she will soon become concerned, if her project is successful, about how to manage the stream of contributions and maintain a good, collaborative ambiance for development. This means open source organization are frequently designed primarily to screen and filter contributions (von Krogh, Spaeth, and Lakhani 2003).

2 When to choose the BSD license and the hijacking problem

Consider the case where $\delta = 0$ and $F = 1$. There is one feature to be developed and there is only one round of development.

Proprietary license terms Suppose the project leader chooses proprietary license terms. When the software is developed, the project leader will sell her software at price $p = v$, the value of the software, to every M potential users of the software. She will incur cost c to develop the software and cost c' to make it proprietary.

Lemma 1 *If the proprietary license terms are chosen, the project leader sells the developed software at price $p = v$. The project leader's utility is $\Pi_P = (M+1)v - c - c'$.*

Proof. The project leader maximizes profit

$$\Pi_P = Mp + v - c - c' \tag{3}$$

s.t.

$$v - p \geq 0 \tag{4}$$

Profit is the market price of the software times the number of buyers plus the private utility of the software to the PL minus cost of development c and the organizational cost c' of choosing a proprietary license. The constraint ensures consumers buy the software. It will be saturated so that $p = v$. ■

GPL license terms Suppose the software is licensed under the GPL. Once the software is developed, its price is 0 because the license prohibits the software being made proprietary.[1] Suppose a developer is chosen into the set N of those n developers

contacted to work on the software. If $n = 1$ and $v - c > 0$, then the developer develops the software with probability 1; if she didn't, then the software would never be developed. If $n > 1$ and $v - c > 0$, then a developer will attempt to free ride on the effort of another developer by adopting a mixed strategy and working with probability less than one. The overall probability the software is developed is then less than one. If $v - c < 0$, then no developer works on the software. This brings the following lemma:

Lemma 2 *If the GPL license terms are chosen, $\delta = 0$ and $F = 1$, the project leader discloses the code to one agent. The agent works with probability $x = 1$ for $c < v$ and $x = 0$ for $c \geq v$. The price of the software is 0. The project leader's expected utility is $\Pi_{GPL} = xv$.*

Proof. A general proof for any $\delta \geq 0$ is provided in appendix A. ■

BSD license terms Suppose now the project leader chooses the BSD license terms. In that case, developers are more motivated to work than in the GPL case: They bet on the possibility that they will be the only one to develop the software and to be able to exploit a monopoly on the software. Unlike in the GPL case, therefore, the project leader will not want to disclose the software to only one developer each period. That developer would have a monopoly on the software if she developed it. She would then be able to sell it at price $p = v$ and derive total net profit $Mv - c' + v - c \geq 0$ from its development. The PL's utility would be $v - p = 0$, which is less than what she would get by developing the software herself under proprietary license terms. The project leader therefore chooses to release the code to more than one developer each period, so as to, with positive probability, not have to pay for the software. Indeed, if two developers or more developed the software, it is not possible for any of them to make any profit by making the software proprietary. If one did so and set a price such that she made positive profit, another would make its own software proprietary

and undercut the price of the first. Denote x_n the probability a developer works when $Card(N) = n$.

Lemma 3 *If the BSD license terms are chosen, $\delta = 0$ and $F = 1$, the project leader discloses the code to $n = 2$ agents. The price of the software will be $p = v$ if only one developer develops the software, 0 else. The probability a developer develops the software is $x_2 = 1$ for $c \leq v$, $x_2 = \frac{Mv - c' + v - c}{Mv - c'}$ for $c \in [v, (M+1)v - c']$ and $x_2 = 0$ for $c \geq (M+1)v - c'$. Profit for the PL is $\Pi_{BSD} = x_2^2 v$.*

Proof. See proof in appendix B. Sudipto Bhattacharya proved a similar type of result in an auction setting (see Spatt (1988)). ■

Note how the results differ from those in the GPL case: for $v \geq c$, even though the PL discloses the code to two agents at once, they both develop it with probability one whereas if the code had been released under the GPL, each developer would have developed the code with probability less than one. This is because under the GPL, if one agent develops the software, it must release it for free to others. If two developers are contacted, then they will try to free-ride on each other's effort. Under the BSD on the other hand, agents know that if they do not work and the other agent develops the software, the other agent will sell it at price v. The alternative to working is thus 0 and they therefore both work with probability one when $v \geq c$. Note that if the PL had contacted three agents, then each agent would work with probability less than one. Indeed, they would bet on the possibility that the two other agents develop the software and it thus be released for free. The alternative to working would then be $x_3^2 v > 0$ so that $x_3 < 1$. Therefore, $n = 2$ is optimal for $v \geq c$ as well as for $v \leq c$.

The results also differ from the GPL results for $v \leq c$: in that case, the software would not be developed under the GPL as the private incentives for development would not be sufficient. However, since $Mv - c' > 0$ (assumption 1), a developer may work on the software in the hope that net expense $v - c$ will be recouped by selling

the software and getting its market value. However, each developer will work with probability $x_2 < 1$ because, unlike in the case where $v \geq c$, the benefit of working is not always positive: if the other developed the software as well, then each developer makes a loss of $v - c$.

The choice of the project leader The comparison between the BSD, the GPL and the proprietary license terms can now be made. The GPL is equivalent to the BSD for any $v \geq c$ as under both licenses development will occur with probability one and the PL's utility will be v. Note however that the BSD requires that two developers be contacted while the GPL needs only one. This difference will be discussed later in the article. The BSD will be preferred to the GPL for any $v \leq c$ as the GPL is not feasible in that range. For $v \leq c$, the BSD may be preferred to the proprietary license terms when $x_2^2 v$ the PL's utility under the BSD is more than $Mv - c' + v - c$, its profit under proprietary license terms. As x_2 goes to zero when $Mv - c'$ diminishes, the proprietary license terms will become preferred for $Mv - c'$ low. If $Mv - c'$ is high, then the proprietary licenses become preferred too, as proprietary profit increases while BSD profits cannot go higher than v. It is therefore for intermediate values of $Mv - c'$ that the BSD will be preferred to proprietary license terms.

Proposition 1 *The BSD license terms will be chosen when s.t.* $v \in [\frac{4}{5}c, c]$ *and* $Mv - c' \in [v(\frac{1}{2} - \sqrt{\frac{5}{4} - \frac{c}{v}}), v(\frac{1}{2} + \sqrt{\frac{5}{4} - \frac{c}{v}})]$. *The GPL or the BSD will be chosen s.t.* $c \leq v$ *and* $Mv - c' \leq c$.

Proof. Profit under the proprietary license terms is $\Pi_P = Mv + v - c - c'$ for $Mv + v - c - c' \geq 0$, 0 else. Profit under the BSD is $\Pi_{BSD} = v$ for $c \leq v$, $\Pi_{BSD} = v[\frac{Mv+v-c-c'}{Mv-c'}]^2$ for $c \in [v, (M+1)v - c']$ and $\Pi_{BSD} = 0$ else. Profit under the GPL is $\Pi_{GPL} = v$ for $c \leq v$, 0 else. Straightforward calculation obtains the result.

■

The proprietary license terms are the only feasible for $v \leq \frac{4}{5}c$, will be chosen when $Mv - c' \geq v(\frac{1}{2} + \sqrt{\frac{5}{4} - \frac{c}{v}})$ or $Mv - c' \in [c - v, v(\frac{1}{2} - \sqrt{\frac{5}{4} - \frac{c}{v}})]$ for $v \in [\frac{4}{5}c, c]$ and when $Mv - c' \geq c$ when $c \leq v$.

This part underlined the conflict of interest between the project leader's ideal license terms and what developers prefer. Developers prefer the BSD to the GPL because the BSD allows them to commercialize the result of their work on the software. Users prefer the GPL as it guarantees the software will be available for free. The project leader is indifferent between the GPL and the BSD whenever both are feasible. Under the GPL, she will get any implementation for free while the BSD could allow for a proprietary exploitation of the BSD software that would carry a positive price. By exploiting competition between developers, the PL can get the software for free even when it is under the BSD, and this with probability one whenever $v \geq c$.

For $v \leq c$, developers working under the GPL would lack the motivation to work on the software because there is no way to commercially exploit it. The PL may therefore choose the BSD license term as this will enhance developers' incentives. The PL contacts many developers at once in the hope of generating competition between them and thus lowering the probability the resulting development is made proprietary and carries a positive price. However, as the prospect of competition with other developers lowers developers' incentive to work on the software, the PL will still choose to limit the number of developers in the development team.

As for proprietary license terms, they ensure development occurs because the project leader is in control of development. However, higher organizational costs (cost of protecting the work, of hiring developers and of monitoring them) may lead her to choose an open-source license as an easy way out of the complications of commercial licensing.

A practical conclusion from this part is that the ratio between the cost of development c and the value of the software v may be lower in GPL than in BSD or proprietary

projects.[2] The GPL will be chosen in areas where a small effort brings big rewards. This is the case of less innovative software development areas, such as projects that consist in reverse engineering existing proprietary software. Linux is a good example, as it was originally a UNIX clone which got progressively improved into a viable alternative to the Microsoft Windows system. The BSD-Unix system, on the other hand, was developed from scratch by academics.

From this part too, the number of contributors in a BSD projects (the PL and two developers) should be higher than in GPL projects (the PL and one developer), which itself should be higher than in proprietary projects (the PL develops alone). This can be interpreted in several ways: this can mean that the process of development will be more competitive in BSD projects than in GPL or proprietary projects (as competition is what guarantees that development can keep being done in an open source way). It can also mean that a project leader will choose the BSD only in areas that will attract a lot of interest by developers, while the GPL will be chosen in areas where there is less prospect of an healthy competition between developers. Lerner and Tirole (2005b) do indeed find that the higher the value of the software to the users compared to its value to the developers, the more restrictive the license terms will be.

As the potential market for a software project increases, BSD or proprietary license terms will become more frequently used. It is indeed quite seldom that OS applications are user-oriented rather than developer oriented, and most software with an user orientation are proprietary. The Lerner and Tirole (2005b)'s study would seem to contradict this finding (the GPL is more frequently used the higher the project's value to end-users) but that study also asserts that the BSD is more frequently used the higher the project's value to developers. That study also does not include proprietary software in its sampling since it was based on an OS repository. Verifying the predictions from the present model would require building a fully representative sample of software in an application domain. This is a project I am currently leading.

3 Streams of innovations and forking

The previous part considered there was only one task to be performed. However, software development is generally an incremental process; developers use each other's work to develop increasingly sophisticated products. This part considers the dynamics of incremental development.

Suppose thus that there are two tasks to be performed successively ($F = 2$). In each period, a task f is allocated to n developers. The two tasks are assigned in succession; task 2 cannot be performed if task 1 was not performed. Developers observe the outcome of the previous development rounds and they know how many development rounds there are.

In such a setting, the PL faces the risk that the whole project may be appropriated in the first phase of development, which may make her less willing to choose the BSD.

That fear is irrelevant in the case of the GPL and the proprietary license terms as each feature is developed with probability one each period and the development thus progresses unimpeded. In the BSD case, it is only when the two developers developed the feature 1 that OS development will be able to proceed to feature 2. Indeed, in that case, both developers release the code embodying feature 1 to the community under the BSD since none gained an advantage over the other. When only one developed feature 1, then she will make it proprietary and it will not be possible for other developers to proceed further in developing the software. Open-source development will stop because there are no sufficient incentives to replicate the existing proprietary feature open-source: it would cost c to develop which is less than v, its private value, and since a proprietary version already exists, there is no perspective of making a proprietary exploitation of it. Finally, if no developer developed the software, then development stops. Once a developer gained an advantage on others and made its software proprietary, she will keep developing the software in a proprietary way and appropriate sur-

plus from feature 2 as well. This means the developers are more motivated to develop the first feature than the second. Indeed, the sooner the software is made proprietary, the higher the reward as the possibility to exploit the sale of future improvements is also acquired.

The following proposition shows that the higher the number of features to develop, the more advantageous the BSD becomes. The project leader may be putting the whole project at risk of early proprietary hijacking, but the probability of early hijacking becomes lower as the project becomes more complex. Developers are indeed more motivated to develop feature 1 rather than feature 2, which means that hijacking will occur more often in the later stage of development (2) than in the early phase (1).

Proposition 2 *The PL's utility from choosing the BSD increases as the project becomes more complex (F increases) because this increases the chances the project will keep being developed under the BSD rather than hijacked. The PL's utility is* $\Pi_{BSD} = (x_2')^2(1+\rho x_2^2)v$ *with* $x_2 = 1 + \frac{v-c}{Mv-c'}$ *and* $x_2' = 1 + \frac{v-c+\rho x_2^2 v}{(Mv-c'+v-c)(1+\rho)-(v-c+\rho x_2^2 v)}$ *for* $c \in [v, Mv+v-c-c']$.

x_2 is the probability the second feature is developed, and is the same as in the case where $F = 1$ as there is only one feature left to develop. x_2' is the probability the first feature is developed, and is higher than x_2 as developers' profit from hijacking is higher when 2 features are left to develop than when only one is.

Proof. In appendix C. This can be generalized to a stream of innovation for any $F > 2$. ■

In this simple model, once a developer gained an advantage at one stage in the development process, that advantage is irreversible and OS development cannot go on. In a more complex model, the open-source version may keep being developed in the hope that at some point it might catch up again with the proprietary project and become potentially marketable. The BSD main line of development would then coexist with

more advanced proprietary versions of the BSD software. This is what happened for example in the case of the BSD TEX typesetting software. However, in the present setting, the developer who gained an advantage at some point develops the following features in succession with probability one in each period so there is no possibility to catch up.[3]

GPLed innovations will cumulate without forking or appropriation by others, while BSD development runs the risk of being hijacked and thus stopped as soon as it is made proprietary. Does this make the GPL preferable to the BSD? The BSD does as good a job as the GPL whenever the GPL is feasible. Indeed, there is no risk of hijacking for $v \geq c$ because the PL contacts two developers and both developers work with probability one, ensuring the resulting work is contributed back under the BSD. The BSD may lead to forking when $v \leq c$ but the GPL would not have been possible under those conditions anyway.

It was assumed that when two developers developed a feature at the same time, they would contribute them back to the common pool and development would keep on from there as if only one version of the feature existed. This supposes developers are able to agree on which version is best, or can reconcile differing versions easily. That may not correspond to reality as ego motivations come into account. Technical disagreements and small differences in the implementation of the same function can also be important. The two versions may then keep on being developed independently, which would substantiate the critique made by GPL proponents: the GPL would be preferable to the BSD whenever both are feasible because it prevents development fragmentation. The advantage of the GPL would be that its development does not rely on competition between developers for its success.

However, there might be a benefit to fragmentation; competing development teams that work on related versions of BSD software might benefit from each other's work.

Fragmentation may be seen as wasteful replication of effort but in fact competition between projects can improve the resulting software, as participants in the projects will try to overtake and improve on each other, possibly borrowing the best from each other. This was certainly what happened in the parallel development of the different strands of BSD Unix as development teams frequently and openly borrowed each other's best features.

Since this part ends up introducing competition issues into the development of BSD software, the next part examines what happens when several projects are put into competition.

4 Competition between projects

Consider two project leaders ("PL") with two projects that are perfect substitutes. The first PL, PL_1, first chooses the license terms under which her code will be developed, contacts n_1 developers who will work under the conditions it chose, and obtains (or not) a product. The second PL, PL_2, having observed that choice, and its result, chooses her own license terms and n_2. The chronology of the development of each project is unchanged from that exposed in part 1. Assume that $\delta = 0$ and $F = 1$.

This part examines which license terms PL_2 will choose in response to the choice of PL_1 and how competition changes the choices of the project leaders.

Lemma 4 *If the first project leader chooses proprietary license terms, then she will face no competition, open-source or otherwise. The price of her software will be* $p = \min[c, v]$ *and the project leader's utility will be* $\Pi_P = Mp + v - c - c'$.

Proof. First note that whether the second PL releases its software under the GPL or the BSD, the incentives of the developers will be the same since when development

occurs, the price p for both software will be lowered to 0 as the two products are perfect substitutes. The choice of the GPL or the BSD is therefore indifferent as developers' incentives are limited to their private value for the software. The first PL will choose her price knowing the second PL will release her code under an open-source license or buy her own product. Suppose $v \geq c$. A developer who works gets $v - c$ while if she doesn't work she gets $\max[v - p, 0]$ so that the first developer can sell only if $v - p \geq v - c$ or $p \leq c$. By maximization of profit, she sets $p = c$. If $c > v$, the proprietary software developer sets $p = v$ and then neither the BSD or the GPL open-source product will get developed. Indeed, the GPL is not feasible, while the incentives to develop the BSD software are reduced to private incentive $v - c$ so that the BSD is not feasible either. ■

This lemma precludes not only the emergence of a proprietary alternative to the first PL's software, but also of any open-source alternative. That the second PL doesn't want to develop her product under proprietary license terms if the first develops her own product is not surprising, as indeed competition would reduce prices to 0. Since price $p = \min[c, v]$, she will rather buy PL_1's software rather than developing her own project herself and incur development cost c. What is more surprising is that the second PL is not tempted to release her code under either the GPL or the BSD, even though that would produce an alternative to PL_1's proprietary software and reduce its price to 0 at no cost to the second PL. This is not done because the first PL anticipates that action and will set her price p equal to c so that even if an alternative OS project was launched, its developers are indifferent between working on the OS product (cost c) or buying the proprietary product (price c)

The only change from the monopoly case is for $c < v$; competition reduces the price of the proprietary product to c instead of v. For $c > v$, the first PL can act as a monopolist.

Consider now the case where the first PL chooses an open-source license. When the first PL chooses the GPL, then there is no point for the second project leader launching her project since the first one will be made available for free to her. The same holds if the first PL chooses the BSD and $v \geq c$. Now, if the first PL chooses the BSD and $v \leq c$, then the second project leader may choose to launch her project if the feature developed in the first project is made proprietary. However, no one will be motivated to work on it since the feature will be sold at price $p = v$ whereas developing the software again would cost $c \geq v$. The second project leader will then choose to launch her own project only if development of the first project was not achieved. This has an ambiguous effect for PL_1: on the one hand, developers in the first project will be tempted to free ride on the effort of developers in the second project. On the other hand, PL_1 will benefit if the second project is successful and is kept BSD as she will then get it for free. The following lemma shows that competition will always make the BSD more attractive to a project leader.

Lemma 5 *If the first project leader chooses open-source license terms, then its project will have no competition. GPL profit is unchanged from the monopoly case. BSD profit for the first PL is* $\Pi_{BSD} = (x_2')^2 v + (1 - x_2')^2 x_2^2 v$ *when* $v \leq c$ *with* $x_2 = \frac{Mv - c' + v - c}{Mv - c'}$ *and* $x_2' = \max[0, x_2 - \frac{x_2^2 v}{Mv - c'}]$, *s.t.* $c \in [v, (M+1)v - c']$. *This is more than profit in the monopoly case.*

x_2' denotes the probability a developer works on the first project and x_2 the probability a developer works in the second project. x_2 is as in the monopoly case, while x_2' will be less than in the monopoly case since there is free-riding by the developers. The PL's profit is $\Pi_{BSD} = x_2'^2 v + (1 - x_2')^2 x_2^2 v$: development of the first project is made by both developers with probability $x_2'^2$ and released for free. If none work (probability $(1 - x_2')^2$), then the second project may be developed and released under the BSD with probability x_2^2. As long as the BSD is chosen by PL_2, x_2 is high enough for x_2' never to get so low that the net effect be negative.

Proof. The proof is in appendix D. The proof can be generalized to a stream of competitors: Since PL_1 welcomes competition for her project, PL_2 will welcome competition to her project as well, and so on ad infinitum. The condition for competition to be welcomed only depends on the relationship between x_2' and x_2, which remains the same as in the lemma. ■

It is now possible to state the following proposition:

Proposition 3 *Competition reduces profits for a proprietary licensor, does not change profits for a GPL licensor and increases profits for a BSD licensor. Competition thus makes open-source licenses more attractive.*

Proof. From lemma 6 and 7 and because p, the price of a proprietary software, decreases compared to the monopoly case in order to deter potential competition. Since GPL profit is unchanged, the range where open source licenses are used is clearly broadened when $v \geq c$ while it is also broadened for $v \leq c$ as PL_1's BSD profits is increased by competition. ■

As explained in lemma 5, the proprietary license terms are still used even in front of a potential open-source equivalent. When the project leader chooses a proprietary software license, she must lower her price so as to deter the development of an alternative open-source software. The potential availability of an open-source alternative doesn't make the use of proprietary license terms disappear. The proprietary license may however be used less often in application fields that are very competitive than in application fields where competition is limited. However, if used, the proprietary software will have only limited OS competition. More generally, GPL and proprietary software will generally gain a monopoly in their domain of application (the first project wins), while the history of BSD software will be more complex: Several BSD projects may be launched in succession and abort until one is successful. BSD contributions will be dispersed over many projects and many projects will die out. This is borne by

the example of how competing set of macros were built on top of TEX under the leadership of different individuals, with only one (Leslie Lamport's LATEX) remaining in the end as less popular projects died out. Many strands of development competed over time, died out or were sometime integrated into the most successful of the alternative development projects (see Gaudeul (2005b)). GPL development on the other hand will appear to be less chaotic.

5 Conclusion

The BSD expands the range of projects that can be developed open-source. It will generally not be chosen in cases where the GPL might have been chosen had the BSD not existed. The BSD and the GPL are therefore not substitutes but rather complements. This finding takes the edge from the GPL vs. BSD controversy, and also defines in precise terms how BSD software will differ from GPL software. GPL proponents' fears of proprietary hijacking of BSD software are justified, but the paper shows this is a conscious risk taken by the PL. She will mitigate the risk of seeing the project hijacked into proprietary versions by contacting many developers at once so there be a high probability competing proprietary versions exist and the price be lowered. Forking will happen more often in BSD projects than in GPL ones but paradoxically, BSD license terms become more and more attractive as a project becomes more and more complex and farther from completion. This is because the probability there is forking is very low at the beginning of the project as developers strive to be the first to make the project proprietary and appropriate it at an early stage. They thus keep each other in check which guarantees that forking will occur only seldom in the early stages of development. There will generally not be competition between GPL, BSD or proprietary versions of a software; a project leader will choose the optimal license for her software and subsequent projects in that same domain of application will not take off, whether

they choose different or the same type of license as the first software in the market. Strong competition will make proprietary licenses less attractive. When the BSD is chosen, projects may emerge and fail until one is successful or made proprietary while GPL software development will be less uncertain.

Future work would concentrate on the robustness of the results; what happens if the production function is changed, when for example there are synergies between developers (team work), or when the OS organization can generate a system of reward and punishment for developers. The literature on public goods, notably on the motivations of contributors in an experimental setting, also shows that contributors have other motivation than their mere individual interests. The paper assumed there was only one possible way to develop the software project, while in reality, even close versions of the same OS product differ in some respect and fit subtly different needs. The paper could explicitly make a difference between the value of the software to developers and to consumers instead of assuming it is the same to both, so as to determine the effect of the user-orientation or developer-orientation of the software as in Lerner and Tirole (2005b). A proprietary version of a software and its OS version were assumed to be equivalent as long as they included the same features. In reality, there are impediments to the direct use of an OS software by lay-users while proprietary software is frequently easier to use. This is why consumers may prefer proprietary software even if it is functionally equivalent to its OS version (Gaudeul (2005a)).

Finally, as the paper predicts development teams will be larger in BSD software than in GPL software (as in BSD software developers prefer to be many so as to lessen the risk the development is hijacked, or, in an alternative explanation, the BSD is chosen in development areas that attract many developers), the leadership and management style of BSD and GPL projects may differ. BSD organizations may need to follow a collegial decision process while GPL organizations would be more likely to be directed from the top under the authority of one leader. This is because BSD developers have

a higher 'out' option than GPL developers, so that the way a decision on development is reached might need to be more consensual. Systematic comparison of the way GPL and BSD projects function could confirm this intuition. For example, while there is an undisputed leader in the development of the GPL Linux operating system, the development of the BSD Apache server is managed through committees and consultation with a broad array of developers.

Notes

[1]Indeed, the software cannot be sold without its source code being included in the product. The first buyer will thus become a competitor to the first seller and after the first sale price will thus drop to 0 through competitive pressure. Every agent will wait for another to buy the software as long as the first seller sets a price more than 0, so that the equilibrium price p is 0. This line of reasoning is similar to the Coase conjecture for the pricing of durable goods. A simpler justification for the 0 price is that the GPL prohibits selling the software being sold at more than a nominal price covering distribution costs.

[2]This is not equivalent to saying that work will be divided into smaller chunks in GPL development than in BSD or proprietary development or to saying that the cycle of releases in GPL software is faster than in BSD software and yet faster than in proprietary software (Evidence from Fershtman and Gandal (2005) is that GPL developers make smaller individual contributions than BSD developers). Indeed, this presumably reduces both c and v in parallel.

[3]The OS strand of BSD development may also keep on because some consumers prefer the OS version even if it is less advanced, but the incentives of OS developers are nonetheless reduced by the coexistence with the proprietary version.

References

BERGSTROM, T., L. BLUME, AND H. VARIAN (1986): "On the Private Provision of Public Goods," *Journal of Public Economics*, 29, 25–49.

FERSHTMAN, C., AND N. GANDAL (2005): "Open Source Projects: Output per Contributor and Restrictive Licensing," Working Paper, Tel-Aviv University.

GAUDEUL, A. (2003a): "The (LA)TEX project: A case study of open-source software," *TUGBoat*, 24, 66–79.

——— (2003b): "The (LA)TEX project: A case study of open-source software," Ph.D. thesis, chapter 3.

——— (2005a): "Competition between open-source and proprietary organizations," CCP Working Paper 05-3.

——— (2005b): "Open-source organizational dynamics and competitive strategies: The (LA)TEX case study.," Working Paper, University of East Anglia.

HANN, I.-H., J. ROBERTS, S. SLAUGHTER, AND R. FIELDING (2004): "An empirical analysis of economic returns to open source participation," Working Paper, Carnegie Mellon University.

HELLMANN, T., AND E. PEROTTI (2004): "The Circulation of Ideas: Firms versus Markets," Working Paper, Stanford GSB and University of Amsterdam.

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

LERNER, J., AND J. TIROLE (2005a): "The economics of technology sharing: open source and beyond," *Journal of Economic Perspectives*, 19(2), 99–120.

——— (2005b): "The Scope of Open-Source Licensing," *Journal of Law, Economics, & Organization*, 21(1), 20–56.

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

PALFREY, T. R., AND H. ROSENTHAL (1984): "Participation and the provision of discrete public goods: a strategic analysis," *Journal of Public Economics*, 24(2), 171–193.

SPATT, C. S. (1988): "Strategic Analysis of Takeover Bids," in *Financial Markets and Incomplete Information*, ed. by S. Bhattacharya, and G. M. Constantinides, pp. 106–121. Rowman and Littlefield Publishers, Inc.

VARIAN, H. R. (2004): "System Reliability and Free Riding," Working Paper, University of California, Berkeley.

VON HIPPEL, E., AND G. VON KROGH (2003): "Open Source Software and the "Private-Collective" Innovation model: Issues for Organization Science," *Organization Science*, 14(2), 209–223.

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.

A GPL license terms

Lemma 6 *If the GPL license terms are chosen,* $\delta \geq 0$ *and* $F = 1$*, the project leader discloses the code to one agent every period until the project is developed. The agent works with probability* $x = 1$ *for* $c < (1-\delta)v$*,* $x = \frac{1-\delta}{\delta}\frac{v-c}{c}$ *for* $c \in [(1-\delta)v, v]$*,* $x = 0$ *for* $c \geq v$*. The price of the software is* 0*. The project leader's expected utility is* $\Pi_{GPL} = \frac{xv}{1-(1-x)\delta}$.

Proof. Denote s_i the strategy by developer i in N. Consider the mixed strategy s_i which consists in choosing $a_i = 1$ with probability x_i. The expected probability the software is developed is then

$$E(P(F(a_1, ..., a_n))) = 1 - (1-x_1)(1-x_2)...(1-x_n) \tag{5}$$

Denote $\widehat{s}_i = BR(s_1, ..., s_{i-1}, s_{i+1}, ..., s_n)$ the best response by i to the strategies of other developers. A Nash equilibrium is such that $\forall i, s_i^* = BR(s_1^*, ..., s_{i-1}^*, s_{i+1}^*, ..., s_n^*)$. Consider only symmetric equilibria of this game and denote s_n the optimal symmetric strategy when there are n developers, x_n the probability each developer will work under s_n and $q_n = 1 - x_n$. $\widehat{s}_i$ the best response to $(s_1, ..., s_{i-1}, s_{i+1}, ..., s_n)$ is defined by

$$\widehat{x}_i = \max_{x_i} \{x_i(v - c) + (1 - x_i)[(1 - q_n^{n-1})v + \delta q_n^{n-1} \frac{1 - q_n^n}{1 - \delta q_n^n} v\} \tag{6}$$

$v - c$ is the net utility of working. $(1 - q_n^{n-1})v + \delta q_n^{n-1} \frac{1-q_n^n}{1-\delta q_n^n} v$ is the utility of not working: $(1 - q_n^{n-1})v$ is the probability another developer in this period develops the software times v, its private value. q_n^{n-1} is the probability no developer works this period so the game is brought forward next period, and discounted by δ. $\frac{1-q_n^n}{1-\delta q_n^n} v$ is the discounted expected utility as n developers will be contacted each period. Maximization in x_i shows that

$$v - c = (1 - q_n^{n-1})v + \delta q_n^{n-1} \frac{1 - q_n^n}{1 - \delta q_n^n} v \tag{7}$$

The project leader thus maximizes profit

$$\Pi_{GPL} = \frac{1 - q_n^n}{1 - \delta q_n^n} v \tag{8}$$

s.t.

$$v - c = (1 - q_n^{n-1})v + \delta q_n^{n-1} \frac{1 - q_n^n}{1 - \delta q_n^n} v \tag{9}$$

Profit is the private utility of the software to the PL times the probability the software is developed each period, $1 - q_n^n$, discounted by $q_n^n \delta$ each period.

From the constraint in the maximization program above, q_n is determined by the equality

$$c q_n^n \delta + v(1 - \delta) q_n^{n-1} - c = 0$$

The derivative of the above with respect to n is also equal to 0 so that we have:

$$(cnq_n^{n-1}\delta + v(1-\delta)(n-1)q_n^{n-2})\partial q_n + (cq_n^n \ln q_n \delta + v(1-\delta)q_n^{n-1}\ln q_n)\partial n = 0$$

$$n(cq_n^n\delta + v(1-\delta)q_n^{n-1})\frac{\partial q_n}{q_n} - v(1-\delta)q_n^{n-2}\partial q_n + c\ln q_n \partial n = 0$$

$$(ncq_n^{-1} - v(1-\delta)q_n^{n-2})\partial q_n + c\ln q_n \partial n = 0$$

so that

$$\frac{\partial q_n}{\partial n} = -\frac{c\ln q_n}{ncq_n^{-1} - v(1-\delta)q_n^{n-2}} \tag{10}$$

and

$$\frac{\partial q_n}{\partial n} > 0$$

if

$$ncq_n^{-1} - v(1-\delta)q_n^{n-2} > 0 \tag{11}$$

But

$$c = v\frac{(1-\delta)(1-x_n)^{n-1}}{1-(1-x_n)^n\delta}$$

so that (11) can be rewritten as

$$n \geq 1 - q_n^n\delta$$

which is true for any $n \geq 1$ as $\delta \leq 1$. Therefore, q_n is always decreasing with n.

The derivative of the PL's profit function with respect to n is

$$\frac{d\Pi}{dn} = \frac{\partial \Pi}{\partial q_n}\frac{\partial q_n}{\partial n} + \frac{\partial \Pi}{\partial n}$$

which sign is the sign of

$$-(1-\delta)q_n^{n-1}(n\frac{\partial q_n}{\partial n} + q_n \ln q_n)$$

Replacing $\frac{\partial q_n}{\partial n}$ with its expression in (10) the sign of the above is the sign of

$$-(1-\delta)q_n^{n-1}\frac{-v(1-\delta)q_n^{n-1}\ln q_n}{ncq_n^{-1}-v(1-\delta)q_n^{n-2}}$$

knowing (11), the sign of the above is the sign of

$$(1-\delta)q_n^{n-1}v(1-\delta)q_n^{n-1}\ln q_n \tag{12}$$

which is negative, so that the profit function is always decreasing with n and the optimal team size is then 1.

This means that

$$\Pi_1 = \frac{x_1}{1-(1-x_1)\delta}v$$

with $x_1 = \frac{1-\delta}{\delta}\frac{v-c}{c}$ for $c \in [(1-\delta)v, v]$, $x_1 = 1$ for $c \leq (1-\delta)v$ and $x_1 = 0$ for $c \geq v$. ■

For c low enough, a developer will work on the project with probability one, so that there is no need to release the code to more than one developer. When c increases, a developer may want to rely on another developer to do the work, either in this period (which can happen if $n > 1$) or in the next period. The developer then chooses a mixed strategy and works with probability x_n, dependent on n. x_n decreases as the number of developers who were contacted increases. As n increases, x_n decreases so fast that $1-(1-x_n)^n$, the probability that the software is developed by at least one developer among n, decreases too: The incentives for developers to free-ride always over-ride the benefit from having a larger team size so that the optimal n is equal to 1. For $c \geq v$, the benefit of developing the software is lower than its development cost, so that the software is not developed.

B BSD license terms

Denote s_i the strategy by developer i in N. Consider the strategy s_i which consists in choosing $a_i = 1$ with probability x_i. The expected probability the software is developed is then

$$E(P(F(a_1, ..., a_n))) = 1 - (1 - x_1)(1 - x_2)...(1 - x_n) \tag{13}$$

The expected probability only one developer develops the software (Card$(N') > 1$) is

$$E(P(F(a_1, ..., a_n))/\exists! i \text{ s.t. } a_i = 1) = \sum_{i=1}^{n} (1-x_1)...(1-x_{i-1})x_i((1-x_{i+1})...(1-x_n) \tag{14}$$

In that case the developer is a monopoly and sets price p for her software.

The probability that more than one developer develops the software (Card$(N') > 1$) is

$$E(P(F(a_1, ..., a_n))/\exists\, i, j \text{ s.t. } a_i = a_j = 1) \tag{15}$$

$$= E(P(F(a_1, ..., a_n))) - E(P(F(a_1, ..., a_n))/\exists! i \text{ s.t. } a_i = 1) \tag{16}$$

In that case, the developers have equivalent software. If more than one made the software proprietary, Card$(N'') > 1$ and price is driven to 0. If only one makes it proprietary (Card$(N'') = 1$), then it must set its price such that another is not tempted to make its software proprietary as well, which means its expected profit is 0. Assume therefore that if Card$(N') > 1$ then Card$(N'') = 0$, i.e. if more than one developed the software, none make it proprietary (alternatively, we could assume that p is set such that $Mp = c'$ which ensures profit from making the software proprietary is 0).

Denote $\widehat{s}_i = BR(s_1, ..., s_{i-1}, s_{i+1}, ..., s_n)$. A Nash equilibrium is such that $\forall i$, $s_i^* = BR(s_1^*, ..., s_{i-1}^*, s_{i+1}^*, ..., s_n^*)$. Consider only symmetric equilibria of this game and denote s_n the optimal strategy when there are n developers, $n > 1$, x_n the probability an individual developer works under that strategy and $q_n = 1 - x_n$. $\widehat{s}_i$ the best response to $(s_1, ..., s_{i-1}, s_{i+1}, ..., s_n)$ is defined by

$$\widehat{x}_i = \max_{x_i} x_i(q_n^{n-1}(Mp + v - c') + (1 - q_n^{n-1})v - c) \tag{17}$$

$$+ (1 - x_i)[\mu(n-1, x_n)v + (n-1)x_n q_n^{n-2}(v - p)] \tag{18}$$

$\mu(n, x_n) = 1 - nx_n(1 - x_n)^{n-1} - (1 - x_n)^n$ is the probability that the software is developed, and this by more than one developer, when the code is released to n developers. Indeed, a developers who works gets $q_n^{n-1}(Mp + v - c') + [1 - q_n^{n-1}]v - c$ (with probability q_n^{n-1} the developer is the only one to have worked on the software, she then has a monopoly on the software, makes it proprietary and will sell it at price p to the M consumers). If she doesn't work, she gets the software for free when more than one among the remaining $n - 1$ developers developed the software.

Maximization in x_i shows that

$$q_n^{n-1}(Mp + v - c') + (1 - q_n^{n-1})v - c = \mu(n-1, x_n)v + (n-1)x_n q_n^{n-2}(v - p) \tag{19}$$

The project leader thus maximizes profit

$$\Pi_{BSD} = \mu(n, x_n)v + nx_n q_n^{n-1}(v - p) \tag{20}$$

s.t.

$$q_n^{n-1}(Mp + v - c') + (1 - q_n^{n-1})v - c = \mu(n-1, x_n)v + (n-1)x_n q_n^{n-2}(v - p) \tag{21}$$

$$v - p \geq 0 \tag{22}$$

The PL's payoff is the probability the software is developed times the private utility of the software to the PL minus the price p to be paid if there is only one developer who developed the software: $\mu(n, x_n)v + nx_n q_n^{n-1}(v - p) = q_n^n v - nx_n q_n^{n-1} p$.

The second constraint ensures the PL is ready to pay p, the price asked by a developer who developed the software.

From the second constraint, $v - p = 0$ (the developer sets the price just so that the PL is ready to pay).

Note that profit for the PL is 0 if $n = 1$ because she will have to pay $p = v$ anytime the software is developed. The maximisation programme for $n > 1$ and $v < c$ is

$$\max \mu(n, x_n)v \tag{23}$$

$$\text{s.t.} \tag{24}$$

$$q_n^{n-1}(Mv - c') + v - c = \mu(n-1, x_n)v \tag{25}$$

with $\mu(n, x_n) = 1 - nx_n q_n^{n-1} - q_n^n$. Let us determine $\frac{\partial q_n}{\partial n}$ from the constraint, which can be rewritten

$$c = q_n^{n-1}[(M - n + 2)v - c'] + q_n^{n-2}(n-1)v \tag{26}$$

The derivative of the above expression yields

$$\begin{aligned} 0 = {} & [(n-1)q_n^{n-2}((M-n+2)v - c') + (n-1)(n-2)q_n^{n-3}v]\partial q \\ & + \ln q[q_n^{n-1}[(M-n+2)v - c'] + q_n^{n-2}(n-1)v]\partial n \end{aligned}$$

or

$$0 = \frac{n-1}{q}[q_n^{n-1}((M-n+2)v - c') + (n-1)q_n^{n-2}v - q_n^{n-2}v]\partial q + \ln q(c)\partial n \tag{27}$$

$$0 = \frac{n-1}{q}[c - q_n^{n-2}v]\partial q + \ln q(c)\partial n \tag{28}$$

so that

$$\frac{\partial q}{\partial n} = \frac{-\ln q \times c}{(n-1)[c - q_n^{n-2}v]}q \tag{29}$$

Now, rewrite

$$\Pi_{BSD} = v - nq_n^{n-1}v + (n-1)q_n^n v \tag{30}$$

then

$$\frac{\partial \Pi_{BSD}}{\partial n} = -q_n^{n-1} + q_n^n + n(n-1)(q-1)q_n^{n-2}\frac{\partial q}{\partial n} + \ln q \times q_n^{n-1}[(n-1)q - n] \tag{31}$$

which sign is the sign of

$$q - 1 - \frac{n(q-1)\ln q \times c}{c - q_n^{n-2}v} + \ln q \times n \times (q-1) - q\ln q \tag{32}$$

$$= (q-1)[\frac{c - q_n^{n-2}v - n(q-1)\ln q \times c + \ln q \times n \times (c - q_n^{n-2}v)}{c - q_n^{n-2}v}] - q\ln q \tag{33}$$

Assume $c - q_n^{n-2}v > 0$ (A). Then the sign of the above is the sign of

$$(q-1)[c - q_n^{n-2}v - nq_n^{n-2}\ln q \times v] - q\ln q(c - q_n^{n-2}v) \tag{34}$$

or

$$(q - 1 - \ln q)[c - q_n^{n-2}v] - (q-1)nq_n^{n-2}\ln q \times v \tag{35}$$

which is negative as $q - 1 - \ln q \leq 0$ for any $q \in [0,1]$. This means that $n = 2$ and therefore, the condition (A) for the validity of the above is that $c - v \geq 0$ which

is verified. For $c \leq v$, then, and as explained in the main text, the PL contacts 2 developers and they both develop with probability one. Check now that when $n = 2$, the expression of x_2 and Π_{BSD} are as in the lemma.

C Streams of innovations

If the PL chooses the GPL or the BSD and $v \geq c$, then development occurs with probability one each period and the project can never be hijacked. The same holds whenever the proprietary license terms are used. The interesting case is therefore when $v \leq c$ and the BSD is chosen. In that case, the PL's program when she chooses the BSD is

$$\max (x_2')^2 v + (x_2')^2 \rho x_2^2 v \tag{36}$$

$$\text{s.t.} \tag{37}$$

$$0 = (1 - x_2')(Mv + v - c - c')(1 + \rho) + x_2'(v - c + \rho x_2^2 v) \tag{38}$$

$$0 = (1 - x_2)(Mv - c') + v - c \tag{39}$$

x_2' denotes the probability a developer develops feature 1 when 2 developers are selected in the set N, while x_2 denotes the probability a developer develops feature 2 when 2 developers are selected in the set N. The optimal decision of the PL is indeed still to contact two developers each period. The first constraint balances not working and then either having to pay to use the software developed by the other developer or not having the project developed, vs. working and getting proprietary profits over the two features when the other developer doesn't work, or the private value from the software if the other developer works, plus the discounted value of having the second feature potentially released for free. The second constraint is the same as in the monopoly case.

Profit under the BSD is thus:

$$\Pi_{BSD} = (x_2')^2(1+\rho x_2^2)v \tag{40}$$

with $x_2 = 1 + \frac{v-c}{Mv-c'}$ and $x_2' = 1 + \frac{v-c+\rho x_2^2 v}{(Mv-c'+v-c)(1+\rho)-(v-c+\rho x_2^2 v)}$ for $c \in [v, Mv + v - c - c']$.

Profit under the BSD when $F = 2$ must be compared to $(1+\rho)x_2^2 v$, the discounted value of getting the two features developed in succession when they are not linked in a project and thus are developed with probability x_2^2 each period. The former will be higher than the later when

$$(x_2')^2(1+\rho x_2^2)v \geq x_2^2(1+\rho x_2^2)v \tag{41}$$

or when $x_2' \geq x_2$ which is always the case.

Note that even though the motivations for developers are increased in the first stage of development, there always will be some probability that the software is hijacked in the first stage, i.e. $x_2' < 1$ whenever $v < c$. Also, the condition for $x_2' \geq 0$ is the same as for $x_2 \geq 0$: releasing the whole blue-print for development does not expand the software feasibility set.

D Competition between projects

Suppose PL_1 chooses the GPL and $v \geq c$. If PL_1 thinks the PL_2 will choose $n_2 = 0$, then she will choose n_1 optimally $(n_1 = 1)$ and development will occur. $n_2 = 1$ and $n_1 = 0$ is also an equilibrium of the game. For $v \leq c$, the GPL cannot be chosen by either of the PL.

Suppose now PL_1 chooses the BSD and $v \leq c$. Suppose PL_1 chooses $n_1 = 2$. If the software is developed by only one developer, then it is made proprietary at price $p = v$

and developers contacted by the PL_2 will not be motivated to work on the project as price would then be reduced to 0 and their net profit would be $v-c \leq 0$. If the software is developed by more than one developer, then its price is 0 and there is no point for the PL_2 launching her project. If the software is not developed by any developer then the second PL will release the code to $n_2 = 2$ developers.

The first PL's profit is then

$$\Pi_{BSD} = (x_2')^2 v + (1 - x_2')^2 x_2^2 v \tag{42}$$

with x_2 and x_2' determined by the following equation for $v \leq c$:

$$(1 - x_2')(Mv - c') + v - c = x_2^2 v \tag{43}$$

$$(1 - x_2)(Mv - c') + v - c = 0 \tag{44}$$

so that $x_2 = 1 - \frac{c-v}{Mv-c'}$ and $x_2' = x_2 - \frac{x_2^2 v}{Mv-c'}$.

PL_1's profit is more than profit if there hadn't been competition if

$$(x_2')^2 v + (1 - x_2')^2 x_2^2 v \geq x_2^2 v \tag{45}$$

or

$$\frac{x_2'}{2 - x_2'} \geq x_2^2 \tag{46}$$

Denote $Mv - c' = b$. This translates in

$$\frac{1 - x_2 \frac{v}{b}}{2 - x_2 + x_2^2 \frac{v}{b}} \geq x_2 \tag{47}$$

which can be reduced to the condition that

$$1 - \frac{v}{b}x_2 \geq 0 \tag{48}$$

But the condition for the BSD to be chosen by PL_2 vs. proprietary license terms is that

$$x_2^2 v \geq b + v - c \tag{49}$$

which can be rewritten as

$$(1 + \frac{v-c}{b})^2 v \geq (1 + \frac{v-c}{b})b \tag{50}$$

which is equivalent to (48). This means that whenever the BSD is chosen by PL_2 (who faces no subsequent competition), competition increases profits for PL_1 who will therefore herself also choose the BSD rather than the proprietary license terms.