# Mechanism Design to Promote Free Market and Open Source Software Innovation

Geoffrey Parker and Marshall Van Alstyne
Tulane University and University of Michigan

Preliminary:
Please contact authors for up-to-date version before citing

12-January-2004

Recent developments have challenged one prevailing interpretation of the idea that proprietary systems, enshrined in copyright, create the greatest value. The challenge appears at one level among economic strategists who assert that the greatest value in information goods is not created by the strongest and most restrictive intellectual property protection and in another form by the proponents of Open Source Software who argue for value created by peer review and openly modifiable shared code. We articulate a balance of incentives as indexed by the length of time that software remains proprietary, and openness as indexed by the amount of the platform code base that an author releases to the developer community (and users) to promote the creation of new products. We analyze the trade-off between early and late release based on two novel approaches. The first is a two-sided network externality that explores how the release of free information benefits those who develop as well as those who consume. The second is a framing innovation that places existing licenses in a space that suggests where unexplored socially optimal licenses might exist. Neither technique requires the other and the contribution of each can stand on its own. The combination, however, offers the potential for advancement in a debate where many important trade-offs are often omitted to make analysis tractable.

This work has been supported by National Science Foundation Grant IIS-0338662.

# 1 Introduction

Intellectual property law embodies the principle that incentives lead to the creation of social wealth by granting temporary monopolies to authors and inventors as a stimulus to innovation. Recent developments, however, have challenged one prevailing interpretation of this idea that proprietary systems, enshrined in copyright, create the greatest value. The challenge appears at one level among economists who assert that the greatest value in information goods is not created by the strongest and most restrictive intellectual property protection (Shapiro & Varian, 1999). It appears in another form among proponents of the Free Software Foundation, who argue for openness as a right, and the Open Source Software movement, who argue for value created by peer review, reuse and unfettered redistribution.

This research articulates a balance of incentives and openness that promotes both the genesis of feature rich new software offset by widespread distribution and network externality benefits from open access. The proposed mechanism is quasi-open—adjustable between fully open and fully closed. It behaves as fully open in the sense that part of a platform is (1) freely available, (2) open to inspection, (3) modifiable, and (4) redistributable. This generates network externality and access benefits. It behaves as proprietary in the sense that any modified or derivative works, if offered for sale, are subject to disclosure and royalty requirements. This restores profits and incentives. Like standard open source software, derivative works are subject to the four properties of openness noted above, but unlike standard open source software, these requirements do not bind for a brief proprietary period,

specified by the platform author, during which any $3^{rd}$ party developer may exercise pricing power based on the value of an innovation.

In practical terms, this framework generalizes a range of alternative technologies: fully closed systems such as those governed by restrictive "end user license agreements" (EULAs), partially open systems such as those exposing "applications programming interfaces" (APIs), and even fully open systems such as games where platform adoption is subsidized and profits derive primarily from royalties on the sale of third-party enhancements.

The intuition for this model proceeds specifically from software—and thus operates primarily under copyright—but it applies potentially to any innovation with two properties: (i) the intellectual property represents a platform on which subsequent innovation depends and (ii) property rights can be used to govern the terms of access and subsequent disclosure of derivative works. Both properties exist for operating systems, gaming platforms, audio and visual platforms, and any form of software that allows 'plug-ins.' To some extent, however, these properties also exist for hardware platforms, semiconductor masks, telecommunications infrastructure, and biotechnology.

In the context of patents, related literature examines the question of optimal patent length and breadth, finding that property rights should be long-lived (infinite) and narrow when the flow rate of monopoly profits causes welfare losses (Gilbert & Shapiro, 1990; Klemperer, 1990). Alternatively, they should be short-lived and broad when consuming inferior substitutes is the principal cause of welfare losses (Klemperer, 1990). In contrast, our work considers the original innovator's

interest in subsequent developments, a factor that leads to finite-lived protection. Our mechanism places the original author in a position analagous to that of a social planner. And while a social planner chooses a shorter proprietary period than the platform author, the author substantially internalizes the social gains from innovation and therefore chooses a horizon on first round innovation that brings forward the profits from second round innovation.

This work complements the literature on sequential innovation, which argues for longer property rights when innovation proceeds in rounds led by different firms (Green & Scotchmer, 1995). If property rights are short, firms appropriate little of the social value of derivative innovations and underinvest in basic research. Our work takes the next practical step and asks, given a platform and potential for sequential innovation, what mechanism design maximizes profits and welfare. If network effects are small or the ability to build on the platform is small, then the author prefers to go it alone. If, however, either network effects or the coefficient of reuse are large, then the author prefers to engage complementary investment via opening the platform even at the cost of losing profits on the platform itself.

Openness of the platform, coupled with openness of derivative works upon expiration of the proprietary period, offers benefits analogous to those of patent pools. With cumulative innovation, strengthening property rights in mutually blocking technologies can have the perverse effect of stifling innovation (Shapiro, 2001). Firms that cross-license and make the collection of property rights available as a pool avoid this and other difficulties (Shapiro, 2001; Rey & Tirole, 2003). Relative to a patent pool, the open platform and subsequently open derivative works also allow users

and developers to treat the asset as common resource. It reduces problems of multiple marginalization; and it limits transaction costs of negotiating in the context of multi-party hold-up. Openness with default access also avoids problems of exclusive dealing and cartels that are downside risks of patent pools.

A commitment to open the platform, and keep it open into perpetuity, borrows insight from the second-sourcing literature (Farrell & Gallini, 1988). If users incur high asset-specific setup costs, and buy goods either in multiple periods or as components of a system, then a monopoly provider of a platform technology benefits from forward competition. User willingness to adopt ex ante is conditioned by commitments to forgo exploitation ex post. Thus a commitment to an open architecture, or to license to competitors, provides a means of growing the user base when gouging buyers would have been profitable.

The question of how much open code to release and also the conditions for downstream use of intellectual property are decisions that authors can make when designing more successful information goods. At time zero, authors can choose to release some fraction of their code base in order to foster immediate adoption and allow innovation on their platform. They can also set a future date at which all of a code base is released. However, code that is immediately released does not contribute directly to the author's profits. Moreover, promoting subsequent innovation via $3^{rd}$ party enhancements faces the same trade-off: Tension arises between promoting network growth by earlier release and promoting innovation by later release.

## 2 Theoretical Foundations

We analyze the trade-off between early and late release based on two novel approaches. The first is a network externality twist that explores how the release of free information benefits those who develop as well as those who consume. The second is a framing innovation that places existing licenses in a space that suggests where unexplored socially optimal licenses might exist. Neither technique requires the other and the contribution of each can stand on its own. The combination, however, offers the potential for advancement in a debate where many important trade-offs are often omitted to make analysis tractable.

The network externality change is grounded in a two-sided network effect that builds upon Katz and Shapiro (1985). Although a recent area of study, it has begun to receive considerable attention (Parker & Van Alstyne, 2000; Caillaud & Jullien, 2001; Rochet & Tirole, 2003; Armstrong, 2002; Parker & Van Alstyne, 2002). This represents a demand economy of scale that crosses markets as distinct from one that stays within the same market. The presence (or absence) of each side makes the other more (or less) valuable to an organization that deals with both halves at once. Here, the author of an original software platform can look to consumers and other developers as these two halves.

In the present context, author $A$ develops a platform on top of which software developers $D$ can create enhancements that add value for end-users or consumers $C$. Author $A$ then chooses an amount $\sigma$ of platform code to open in order to stimulate developer participation. The author also chooses a time $t$ after which

developer enhancements must be folded back into the open code base. The more that developers are motivated to create enhancements, the greater the benefits to $C$. Similarly, the larger the $C$ market, the more attractive it is for developers to code for that market, contingent on their access to source code. This is consistent with the size of a problem scope influencing the prestige associated with solving it (Raymond, 2000). The computer games market represents another example where the presence of a large number of free game levels makes it attractive to consume particular games; and the presence of a larger gaming user community makes it more attractive to develop for a particular game. Similarly, for operating systems, the larger the installed base of users, the more attractive is developing applications for that OS; and the more applications, the more attractive it is to consume that OS.

# 3 Model

To capture the tension between early and late release, we begin with a simple two period model that includes the value of the platform before developer participation, the first period developer enhancements, and the second round of developer enhancement that builds on the first round after the release period $t$. Timing is as follows:

- Period 0 - Platform author chooses degree of openness ($\sigma$) and downstream developer incentive period $t$
- Period 1 - Developers add value based on $\sigma$ and incentives captured in $t$
- Period 1 - Consumers observe developer value creation and endogenously choose consumption levels
- Period 2 - In a fulfilled expectations equilibrium, developers observe consumer participation in period 1 and choose participation and value-adding levels having gained access to further open code

Figure 1 shows the evolution of value and user participation in the platform.

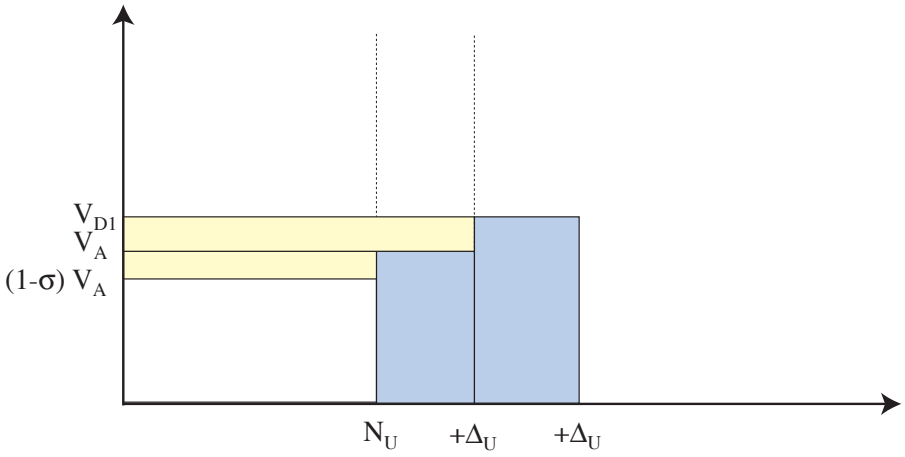


Figure 1: A fraction, $\sigma$, of platform value, $V_a$ is given away, leaving direct platform profit of $(1-\sigma)\,V_a$. However, the release of code allows developer enhancements, which creates value and brings additional consumers to the platform.

### 3.1 Consumer Problem

Consumers know that code will be released under the open regime after period $t$. In a Coasian consumption problem, this conditions their willingness-to-pay in the current period on their discount rate $r$. For tractability, we assume that consumers share a common value for the product. If a developer adds value $v$, this leads

simply to choosing price such that $p \leq v(1 - e^{-r\,t})$. Note that the longer the proprietary period, the more the consumer is willing to pay for a particular developer enhancement. When the proprietary period is very short, the consumer will pay very little since she knows that the product will soon become freely available. In the analysis below, we let $\delta = e^{-r\,t}$.

### 3.2 Developer Problem

We assume that developers produce in direct proportion to the open code base and in response to the length of time they can benefit from their effort. This allows incentive effects to influence code development. In any period, developer output $y$ increases in both $t$ and in the open code base $\Theta$. Let $\Theta_t$ represent open code at time $t, k$ be a re-use coefficient that scales the ability to use open code in new production, and $L(\Theta_t, t)$ be labor or effort as a function of costs and the time to recover investments. Then code created by a developer is $k\,\Theta_t\,L(\Theta_t, t)$. We capture the incentive effect of time by replacing $L(\Theta_t, t)$ with a concave increasing function of time, $(1-\delta)$. The total code created by all of the (symmetric) developers in period $t$ is then $q_{d\,t}\,k\,\Theta_t\,(1-\delta)$. Note that we assume that consumption of one developer's code does not affect the value realized by consuming another developer's code. Hence we assume that the value created by a developer is the same as the amount of code created. That is, $v_t = k\,\Theta_t\,(1-\delta)$. Total value created by all developers is then the same as total code created, $q_{d\,t}\,k\,\Theta_t\,(1-\delta)$. This is consistent with our treatment of code (and value) created by the platform author.

### 3.3 Platform Author Problem

The platform author chooses two parameters. The first is the sharing parameter, $\sigma$, that governs the fraction of code released under open source, which is no longer purchased but is freely adopted by consumers. The second choice parameter is the proprietary time period $t$ that generates revenues for $3^{rd}$ party developers on any enhancements they create. Before $t$ expires, enhancements are profitable but consumer adoption is also limited via positive prices. Upon expiration of $t$, developers contribute derivative works back to the open code base from which they drew their original sources. We explore optimal choices for $t$ and $\sigma$ based on different welfare maximizing criteria, including combinations of freedom of access, adoption rates, and revenues. Since consumers face a Coasian consumption tradeoff, the original firm must factor consumers' strategic behavior into determining the optimal time to release the code under open code terms. Forcing early release of enhancements creates consumer surplus at the cost of reducing developer incentives.

## 4 Analysis

Our point of departure is to make explicit the chicken and egg relationship between customers and developers. To model this relationship, we use a two-sided network externality framework. The network externality term $e_{du}$ measures how much effect the presence of developers has on the size of the consumer market. Conversely, $e_{ud}$ determines how much effect the presence of consumers has on the size of the developer market.

### 4.1 User and Developer Participation

The logic to determine the number of users and developers is that the more users who join the platform, the more developers want to produce content for the platform, and hence the more users who want to join. This feedback generates an infinite sequence that defines the externality between users and developers. To anchor the sequence, let $N_u$ be the core market for the base platform before any developer enhancements are added. The first term of the infinite sequence is then the increase in the number of developers as a function of the number of users, the open code base, and time: $e_{ud}\, N_u\, \Theta$.

The second term is the increase in the number of users as a function of developers, code base, and time: $e_{du}\, e_{ud}\, N_u\, \Theta$. The third term is again the increase in the number of developers as a function of the number of users, the open code base, and time: $e_{ud}\, e_{du}\, e_{ud}\, N_u\, \Theta$.

The total increase in the number of developers can then be found by summing up the odd terms: $e_{ud}\, N_u\, \Theta\, (1 + e_{ud}\, e_{du} + (e_{ud}\, e_{du})^2 + (e_{ud}\, e_{du})^3 + ...)$. So long as $0 \leq e_{ud}\, e_{du} < 1$, this converges to $N_u\, \Theta\, (\frac{e_{ud}}{1 - e_{ud}\, e_{du}})$. Similarly, the total increase in the number of users can be found by summing up the even terms. With the restriction that $0 \leq e_{ud}\, e_{du} < 1$, this converges to $N_u\, \Theta\, (\frac{e_{ud}\, e_{du}}{1 - e_{ud}\, e_{du}})$. Letting $M_u = \frac{e_{ud}\, e_{du}}{1 - e_{ud}\, e_{du}}$ and $M_d = \frac{e_{ud}}{1 - e_{ud}\, e_{du}}$, we define the total number of users and developers as

$$q_u = N_u + M_u N_u\, \Theta \tag{1}$$

$$q_d = N_d + M_d N_u\, \Theta. \tag{2}$$

In the two period model, quantities are subscripted $q_{u1}\,, q_{u2}\,, q_{d1}\,, q_{d2}$ and the open code base is subscripted $\Theta_1$ and $\Theta_2$. For completeness, we have included the term, $N_d$, to represent the number of developers who would participate in the platform independent of any profit motivation. This will facilitate a later analysis of developers who participate in open source projects without any expectation of monetary reward. However, in the first part of the analysis below, we consider those developers who are profit-motivated and thus let $N_d \to 0$.

### 4.2 Open Code Base

In the two period model we analyze, the open code base evolves as follows. In the first period, the platform author can choose what proportion, $\sigma$ to release immediately. This open code base provides the foundation upon which developers are able to create new additions to the platform. In the second period, in addition to the code released in period one, there is new code that was created by the developer community in the first period and then released after the proprietary period, $t$. The developers' ability to reuse code is captured by the parameter $k$. The total amount of code created increases in the total number of developers who join the platform in the first period. As noted above, developer output increases in the length of time over which developers can charge users for enhancements.

$$\Theta_1 = \sigma\, \pi_a \tag{3}$$

$$\Theta_2 = \Theta_1 + k\, q_{d1}\, \Theta_1\, (1 - \delta) \tag{4}$$

In order to facilitate the analysis of welfare and profit as a function of time, we introduce a useful transformation between variables and their first derivatives with respect to time.

**Lemma 1** *The time derivative of second period quantity (for both users and developers) is the difference between first and second period quantity times* $\frac{r\,\delta}{1-\delta}$ *. That is,* $q_2' = \frac{r\,\delta}{1-\delta}(q_2 - q_1)$*. Because* $q_2 - q_1 \geq 0$*,* $q_2' \geq 0$*. Identical to quantity, the time derivative of the second period open code base is also the difference between first and second period code base times* $\frac{r\,\delta}{1-\delta}$*. That is,* $\Theta_2' = \frac{r\,\delta}{1-\delta}(\Theta_2 - \Theta_1)$*.* $\Theta_2 - \Theta_1 \geq 0$ *implies* $\Theta_2' \geq 0$*.*

***Proof.*** *Follows directly from definitions of* $q_{d2}$*,* $q_{u2}\,\Theta_2$*, and* $p$*.* ■

To facilitate the analysis of second order conditions, we introduce a second useful lemma that gives simple transformations between variables and their second derivatives with respect to time.

**Lemma 2** *The second time derivative of second period quantity (for both users and developers) is the difference between first and second period quantity times* $-\frac{r^2\,\delta}{1-\delta}$*. That is,* $q_2'' = \frac{-r^2\,\delta}{1-\delta}(q_2 - q_1)$*. Similarly, the second time derivative of* $\Theta_2$ *is* $\Theta_2'' = \frac{-r^2\,\delta}{1-\delta}(\Theta_2 - \Theta_1)$*.*

***Proof.*** *Follows directly from definitions of* $q_{d2}$*,* $q_{u2}\,\Theta_2$*, and* $p$*.* ■

### 4.3 Welfare Analysis

Total welfare in the two periods is made up of author profit, developer profit, and consumer surplus. The second period is discounted by factor $\delta$ back to the first

period. Detailed expressions for the components of each are given below.

### 4.3.1 Author Profit

$$
\begin{aligned}
q_{u1}\,(1-\sigma)\,\pi_a &= \text{Per 1 author profit, platform sales} && (5)\\
\delta\ (q_{u2}-q_{u1})\ (1-\sigma)\,\pi_a &= \text{Per 2 author profit, platform sales} && (6)\\
\phi\,p_1\,q_{d1}\,q_{u1} &= \text{Per 1 author profit, license fees} && (7)\\
\delta\,\phi\,p_2\,q_{d2}\,q_{u2} &= \text{Per 2 author profit, license fees} && (8)
\end{aligned}
$$

The first term of author profit multiplies the number of users times the amount of value given away by the platform author. The second term, discounted back to period one, multiplies incremental users times fraction of value given away by the platform author. The third and fourth terms are the fraction, $\phi$, of developer sales to consumers that is paid by the developer to the platform author as a license fee.

### 4.3.2 Developer Profit

$$
\begin{aligned}
(1-\phi)\,p_1\,q_{d1}\,q_{u1} &= \text{Per 1 developer profit} && (9)\\
\delta\,(1-\phi)\,p_2\,q_{d2}\,q_{u2} &= \text{Per 2 developer profit} && (10)
\end{aligned}
$$

Total developer profit is made up of sales of developer enhancements to consumers less the license fee paid to the platform author.

### 4.3.3 Consumer Surplus (CS)

$$
\begin{aligned}
q_{u1}\,\sigma\,\pi_a &= \text{Per 1 CS, platform consumption} && (11)\\
\delta\,((q_{u2}-q_{u1})\ \Theta_2 + q_{u1}\,(\Theta_2-\sigma\,\pi_a)) &= \text{Per 2 CS, platform consumption} && (12)\\
(v_1-p_1)\,q_{d1}\,q_{u1} &= \text{Per 1 CS, developer adds} && (13)\\
\delta(v_2-p_2)\,q_{d2}\,q_{u2} &= \text{Per 2 CS, developer adds} && (14)
\end{aligned}
$$

The first term of consumer surplus measures the surplus from consumption of the

portion of the platform that is given away. The second term measures consumption of second period open code $\Theta_2$ by incremental users, $q_{u2} - q_{u1}$, plus consumption of incremental open code, $\Theta_2 - \sigma\,\pi_a$, by first period consumers.

### 4.3.4 Total Welfare

The expression for total welfare is simpler than the elements above might suggest because wealth transfers have no effect and all $\sigma$ and $(1-\sigma)$ terms, and $\phi$ and $(1-\phi)$ terms collapse together. Total welfare can be expressed as

$$q_{d1}\,q_{u1}\,v_1 + \delta\,q_{u2}(q_{d2}\,v_2 + \Theta_2) + \pi_a\,(q_{u1} + \delta\,(q_{u2}\,(1-\sigma) - q_{u1})). \tag{15}$$

The first term is period one developer additions. The second term is second period developer additions plus the value of open code. The third term is first period consumer surplus from the platform (which combines author profit and consumer surplus) plus second period new user surplus less period one double-counting.

Figure 2 shows that the social planner prefers to open up the platform, but prefers an intermediate value for time to release of derivative works.

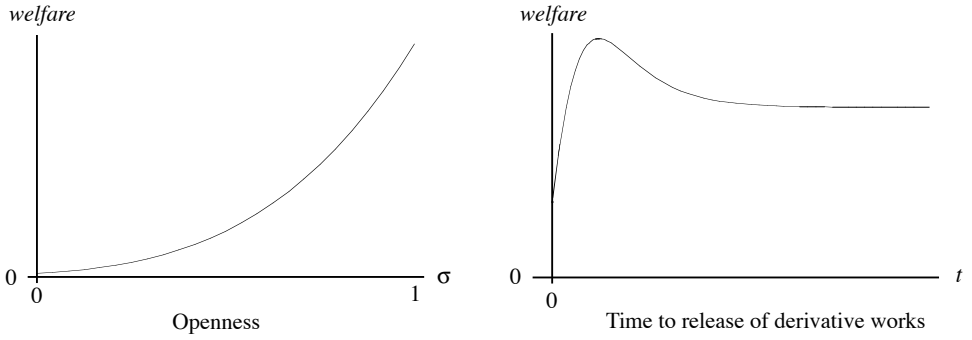


Figure 2: Welfare as a function of time to release of derivative works.

Below, we show that these preferences hold in general when there are positive network externalities.

**Proposition 1** *To maximize welfare in the presence positive network externalities ($M_d, M_u > 0$), the social planner's optimal contract $[\sigma^*, t^*]$ is a corner solution in $\sigma$ but an interior (finite) value in $t$.*

***Proof.*** *Substitute $v_1 = k\,\Theta_1(1-\delta)$, $v_2 = k\,\Theta_2(1-\delta)$, $\Theta_1 = \sigma\,\pi_a$, and $\Theta_2 = \sigma\,\pi_a + q_{d1}\,k\,\pi_a\,(1-\delta)$ into equation 15. After grouping terms, note that each element has a non-negative derivative with respect to $\sigma$, thus establishing the first part of the proposition.*

*To establish the second part, take the derivative of total welfare with respect to time and apply Lemma 1. Note that the derivative of welfare with respect to $t$ at $t = 0$ is positive. The derivative of welfare with respect to $t$ at $t \to \infty$ is zero. As $t$ approaches $\infty$, $\delta$ gets progressively smaller. Ignore higher order terms ($\delta^n$, $n \geq 2$). The sign of the first order $\delta$ terms is negative, showing that the derivative is negative in the neighborhood of $t \to \infty$. Therefore, welfare is falling as $t \to \infty$.* ■

### 4.4 Platform Author Choices

We now explore the platform author's preferences over $\sigma$ and $t$. The platform author is concerned with optimizing the subset of welfare components listed below over both periods. We first establish the importance of network externalities and reusable code to the author's desire to ever open up the platform code base for developers to build upon.

**Proposition 2** *When there are no network externalities between users and developers, such that $M_u$ and $M_d$ are zero, then the platform author never opens the code base and chooses $\sigma = 0$.*

***Proof.*** *After substituting for no network effects, $M_d = 0$, $M_u = 0$ and no developers who work for free, $N_d = 0$, into platform author profit, the resulting expression is $N_u \pi_a (1 - \sigma)$.* ■

### 4.5 Alternative Licenses

A major contribution of this framework is that it generalizes several different types of contracts. We can then ask which contract optimizes a given welfare criterion under a specific set of parameter values. If we allow different degrees of openness $\sigma \in [0, 1]$, we can model a range of access to source code that spans, for example, closed proprietary systems (EULA), access to application program interfaces (APIs), trial-ware, share-ware, royalty free binaries, open source (BSD), and viral open source (GNU). Similarly, different times to release $t \in [0, 1]$ can represent disclosure now, after a brief delay, after a long delay, or on expiration of copyright. Several existing licenses are then captured in Figure 3.

Fully open source software, such as that released under the GNU Public License (GPL) releases everything, $\sigma = 1$, and requires subsequent code to be released immediately, $\delta = 0$. In contrast, the Berkeley Software Distribution (BSD) license releases everything but then places no restrictions on subsequent disclosure, $\delta = 1$. The Microsoft End User License Agreement (EULA) provides no open software, $\sigma = 0$, and therefore places no disclosure restriction on derivative works, $\delta = 1$.

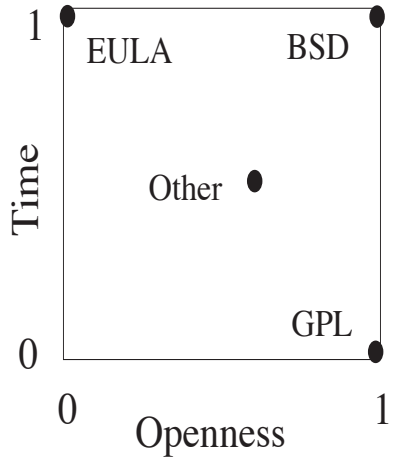


Figure 3: Contracts mapped by time to code release (0=immediate, 1=never) and degree of openness (0=none, 1=complete).

Interestingly, in this framework, many standard licenses appear as corner solutions. Thus we have sought to define and explore licenses with interior solutions for which $0 < \sigma < 1$ and $0 < \delta < 1$. Such a license might be termed a flexible copyright, or meta license, that incorporates features of both proprietary copyright and open source copyleft.

**Proposition 3** *Open BSD and GPL licenses create greater total welfare but unless the royalty rate exceeds* $\frac{1}{M_d\ k\pi_a(N_u+M_u\ N_u\ \pi_a)}$*, a platform author prefers EULA. The relative profits and welfare for EULA, BSD, and GPL style contracts are respectively:*

| *Contract* | *Author Profit* | *Total Welfare* |
|---|---|---|
| *EULA* | $N_u\pi_a$ | $N_u\pi_a$ |
| *BSD* | $M_d\, N_u\, \phi\, k\pi_a^2\, (N_u + M_u\, N_u\, \pi_a)$ | $N_u\ \pi_a\, (1 + M_u\, \pi_a)\, (1 + kM_d N_u\, \pi_a)$ |
| *GPL* | 0 | $N_u\ \pi_a\, (1 + M_u\, \pi_a)$ |

***Proof.*** *These are found by straightforward substitution of corner solutions EULA:* $[\sigma = 0, t = \infty]$*, BSD:* $[\sigma = 1, t = \infty]$*, and GPL:* $[\sigma = 1, t = 0]$ *into the expressions for author profit and welfare.* ■

The open BSD and GPL style contracts always create greater total welfare than the closed EULA license but are not necessarily incentive compatible for the platform author. It is possible that sufficiently large percentage royalties, network effects, or developer value-added make a BSD style contract a dominant author choice although the actual BSD license does not require or even expect royalties; so this framework presents a more generous case from the author's perspective.

It is interesting to note that a BSD contract provides uniformly greater social surplus than the more socially conscious GPL. In fact, the welfare ratio of BSD to GPL is $(1 + kM_d N_u \pi_a)$. This result, however relies critically on the modeling assumption that developers are profit motivated. To probe this further, we examine a model in which developers are motivated by openness $\sigma$ or "freedom".[1] If this is the case, then GPL achieves the highest social surplus of all three licenses. In particular, let $\alpha$ be the fraction of profit motivated developers such that $(1 - \alpha)$ gives the fraction of developers motivated by free access. Then, rewrite the value of developer output $v_t = k\Theta_t q_{dt}(1-\delta)$ to weight effort by relative interest in pricing power and free access. This gives:

$$v_t = k\Theta_t q_{dt}(\alpha(1-\delta) + (1-\alpha)\sigma)$$

[1] Compare, for example, the policy statements found at www.fsf.org.

Under a mix of motivations, the BSD to GPL welfare ratio shifts from $(1 + kM_dN_u\,\pi_a)$ to $\frac{1+M_u\pi_a}{1+M_u\pi_a+kM_dM_uN_u\pi_a}$ as the proportion free access motivated developers rises to the full population. This ratio is clearly $< 1$, showing that a GPL style contract creates greater welfare in this context. Assuming that a social planner must offer a single license and that developers exhibit a mix of motives, it is also straightforward to solve for that fraction of freedom motivated developers such that a social planner prefers to switch from BSD to GPL. Interestingly, if developers are motivated strictly by free access, then a GPL contract is socially optimal not just in relative terms but also absolute terms. That is, welfare exceeds that of a BSD license even when the full population is profit motivated. Setting aside difficult questions on the costs of subsidy and the problem of moral hazard, this advantage parallels certain arguments from the Free Software Foundation made without the benefit of formal analysis.

### 4.6 Private Subcontracting

In addition to the collection of standard open source licenses with $\sigma > 0$, the platform author has the option of subcontracting with specific developers. This can increase profits over those possible from not licensing. There are two primary advantages of subcontracting development relative to open licensing. First, by not opening the code to users, the author need not sacrifice profits on the platform itself. Second, developer contracts might still specify that each must share code with other developers after exploiting the value of their own enhancement. This captures the benefits of a growing code base while allowing developers to charge for the full value

of their individual enhancements.

For comparison, let the value of developer output be verifiable and split based on Nash bargaining. This avoids moral hazard and establishes a more compelling case for subcontracting relative to open contracting.Then, to further increase the comparative advantage of subcontracting, allow second generation output to be available at time 0. This simplifies analysis without affecting incentives, which are no longer based on time. Developers provide two rounds of increased value based on a mutually shared code base that is open with respect to developers but closed with respect to users. Parameters are $t = 0$, $\phi = \frac{1}{2}$, and $\sigma = 1$ but note that there is no profit penalty to the platform author since sharing occurs only privately among subcontractors.

The primary disadvantage of a closed system, however, is that user network effects $e_{ud} = 0$ as closed contracts prevent user access to the underlying code, they disallow modification, and they forbid code sharing. Given these parameters, there still exist regions where a platform author prefers to open the code.

**Proposition 4** *If network effects are absent from either side, such that $e_{du} = 0$ or $e_{ud} = 0$, then subcontracting is always a dominant strategy. In contrast, if network effects are present ( $e_{du} > 0$ and $e_{ud} > 0$), then for all $\sigma$ and $\delta$ in the interval $(0, 1)$ there exist postive constants $\bar{M}_d$, $\bar{M}_u$, $\bar{N}_d$ and $\bar{\pi}_a$ such that if any of $M_d > \bar{M}_d$, $M_u > \bar{M}_u$, $N_d > \bar{N}_d$ or $\pi_a > \bar{\pi}_a$ are true, then the platform author prefers an open license.*

***Proof.*** *Consider first the value of direct platform profits apart from royalties.*

*Substituting parameter values as given for open licensing and closed subcontracting then simplifying the author profit ratio yields:*

$$\frac{2\,(1-\sigma)\;(1+M_u\sigma\,\pi_a\,1+k\,N_d\;\;\delta\,(1-\delta)\;+k\,M_d\,N_u\;\delta\;(1-\delta)\;\sigma\,\pi_a))}{2+k\,N_d\,(2+k\,N_d)}$$

*To prove the first claim, note that either* $e_{du} = 0$ *or* $e_{ud} = 0$ *implies* $M_u = 0$ *, with a ratio of* $\frac{2(1-\sigma)}{2+k\,N_d\,(2+k\,N_d)}$*. Thus no pair of values* $[\sigma, \delta]$ *in the valid interval will suffice to increase the ratio above 1. To prove the second claim, let* $\sigma$ *and* $\delta$ *be any constants in the range* $(0, 1)$ *exclusive (such that* $t$ *lies in the valid interval* $0 < t < \infty$*). Then all terms in the numerator have positive coefficients. Since* $M_d$*,* $M_u$*,* $N_d$ *and* $\pi_a$ *exist only in the numerator, increasing any one value arbitrarily increases the ratio above 1. Analagous results hold for royalties alone but note that adding royalties to the open license platform profits would increase its attractiveness relative to a set of closed subcontracts. Ceteris paribus, the choice of optimal values for* $\sigma$ *and* $t$ *increases this effect relative to arbitrary values.* ■

That rising network effects and user populations should favor open rather than closed licensing is perhaps not surprising. What is more novel is that given any positive level of network effects, a more valuable platform may itself justify giving a fraction away in order to increase author profits.

# 5 Conclusions and Extensions

In this paper, we explore the tension that a platform author faces between fostering platform adoption and follow-on innovation versus immediately capturing the profits to be made from the platform itself. By employing two-sided network externalities, we can explore how the release of free information benefits those who develop as well as those who consume. We consider the welfare of consumers and platform authors, and show that environmental parameters such as the size of the consumer population, the developer pool, the magnitude of externalities in each direction, the ability to reuse code, and the discount rate can affect the optimal choice of time to release and degree of openness.

One contribution of this research is to demonstrate how better licensing can move a monopolist in the direction of a social planner. We show how such a firm benefits by using intellectual property rights to grow their interest in subsequent innovation. Opening a platform increases the attractiveness of third party investment. Controlling the timing of third party disclosure alters both their incentives and also the availability of new code. Longer disclosure times boost incentives; shorter disclosure times open the code stock.

A second contribution is to show how balancing these forces leads to an interior solution in time. Our findings differ from important contributions to the economics of intellectual property in that infinitely long and narrow or infinitely short and broad protections (Katz & Shapiro 1985; Klemperer 1990) need not be optimal. This finding resembles that of Green & Scotcher (1995) in that "standing on the

shoulders of giants" is essential to scientific advancement. That which comes after builds on that which comes before. Where others have shown that shorter term property rights reduce first innovator incentives when different firms undertake a sequence of innovations, we develop a mechanism to facilitate this result.

This is also achieved with a minimum of negotiation via a default contract. Under traditional closed licenses, a third party developer with a good idea might need to negotiate access to source code. Either through negotiation or by observing the identity of the developer, however, the original firm might discern the idea and appropriate it. In practice, large software firms have been accused of this by smaller software firms (Jackson, 1999). This risk reduces their willingness to invest or disclose the idea ex ante. In contrast, a default offer of $\sigma$ and $t$ gives developers an option to enter the market for any fixed costs up to the amount they can recover and without disclosure.

A third contribution is to build a general framework within which to draw comparisons. This framework allows broad-brush welfare analysis of licensing archetypes, including the ability to compare licenses based on developer interest in openness versus profits, and highlights the conditions for optimality.

## References


[1] Armstrong, M. (2002). "Competition in Two-Sided Markets." mimeo: Nuffield College.

[2] Bakos, Y. and E. Brynjolfsson (1999). "Bundling Information Goods: Pricing, Profits, and Efficiency," Management Science 45(12): 1613-1630.

[3] Caillaud, B. and B. Jullien. "Chicken & Egg: Competing Matchmakers." mimeo: University of Toulouse.

[4] Farrell, J. & Gallini, N. (1988) "Second-sourcing as a commitment: monopoly incentives to attract competition" *Quarterly Journal of Economics* 103(4) pp 673-694

[5] Gilbert & Shapiro (1990). "Optimal patent length and breadth" *Rand Journal of Economics* 21(1), 106-112.

[6] Green, J.R. & S. Scotchmer (1995). "On the division of profit in sequential innovation," *Rand Journal of Economics* 26(1), 20-33.

[7] Jackson, T. P. (1999). "U.S. v. Microsoft: Findings of Fact." United States District Court for the District of Columbia.

[8] Katz, M. L. and C. Shapiro (1985). "Network Externalities, Competition, and Compatibility." *American Economic Review* 75(3): 424-440.

[9] Klemperer, P (1990) "How broad should the scope of patent protection be?" *Rand Journal of Economics* 21(1), 113-130.

[10] Lessig, Larry (1999) "Code and other Laws of Cyberspace" Basic Books: New York.

[11] Parker, G. and M. Van Alstyne (2000). "Information Complements, Substitutes, and Strategic Product Design." Social Sciences Research Network (November 8, 2000) http://ssrn.com/abstract=249585.

[12] Parker, G. and M. Van Alstyne (2002). "Unbundling in the Presence of Network Externalities." Mimeo.

[13] Raymond, Eric (2000) "The Magic Cauldron" http://hurkle.thyrsus.com/~esr/writings/.

[14] Rey, P. and J. Tirole (2003) "A primer on foreclosre" forthcoming in Handbook of Industrial Organization III (eds.) Armstrong, M & Porter, R.

[15] Rochet, J.C. and J. Tirole (2003). "Platform Competition in Two-Sided Markets." Journal of the European Economic Association.

[16] Shapiro, C. (2001) "Navigating the patent thicket: cross-licenses, patent-pools, and standard-setting" in *Innovation Policy and the Economy*, Vol 1. (eds.) A. Jaffe, J. Lerner, S. Stern; MIT Press.

[17] Shapiro, C.; Varian, H. R. (1999). *Information Rules: A Strategic Guide to the Information Economy.* Boston, MA, Harvard Business School Press.