MARKETING SCIENCE
CELEBRATING 30 YEARS
Vol. 30, No. 6, November–December 2011, pp. 1066–1078
ISSN 0732-2399 | EISSN 1526-548X | 11 | 3006 | 1066
http://dx.doi.org/10.1287/mksc.1110.0669

Competitive Strategy for Open Source Software

Vineet Kumar
Harvard Business School, Harvard University, Boston, Massachusetts 02163, vkumar@hbs.edu
Brett R. Gordon
Columbia Business School, Columbia University, New York, New York 10027, brgordon@columbia.edu
Kannan Srinivasan
Tepper School of Business, Carnegie Mellon University, Pittsburgh, Pennsylvania 15213,
kannans@andrew.cmu.edu

Commercial open source software (COSS) products—privately developed software based on publicly available source code—represent a rapidly growing, multibillion-dollar market. A unique aspect of competition in the COSS market is that many open source licenses require firms to make certain enhancements public, creating an incentive for firms to free ride on the contributions of others. This practice raises a number of puzzling issues. First, why should a firm further develop a product if competitors can freely appropriate these contributions? Second, how does a market based on free riding produce high-quality products? Third, from a public policy perspective, does the mandatory sharing of enhancements raise or lower consumer surplus and industry profits?

We develop a two-sided model of competition between COSS firms to address these issues. Our model consists of (1) two firms competing in a vertically differentiated market, in which product quality is a mix of public and private components, and (2) a market for developers that firms hire after observing signals of their contributions to open source. We demonstrate that free-riding behavior is supported in equilibrium, that a mandatory sharing setting can result in high-quality products, and that free riding can actually increase profits and consumer surplus.

Key words: open source software; product strategy; signaling; game theory
History: Received: October 12, 2009; accepted: July 6, 2011; Eric Bradlow served as the editor-in-chief and John Zhang served as associate editor for this article.

1. Introduction

The $500 billion software market is undergoing a significant transformation as open source software continues to alter the competitive landscape of the industry (*Economist* 2009). Open source software is built through public collaboration. The source code is published openly, and others are permited to enhance it, in contrast to traditional software where firms closely guard a product's source code.[1] However, open source no longer serves only as a substitute for proprietary software—it is increasingly being incorporated directly into a wide range of commercial products (Gartner Group 2008).[2]

Commercial open source software (COSS) firms build commercial products based on adding more features and enhancing the usability of publicly available open source software. The terms that govern open source licenses dictate how modified versions may be distributed.[3] Certain licenses *require* COSS firms to release feature improvements to the public, where competing firms can incorporate them into their own products. Thus, firms are able to free ride on the contributions of other firms, a practice Microsoft CEO Steve Ballmer referred to as "a cancer that attaches itself in an intellectual property sense to everything it touches" (Fuller 2003).

The unique institutional arrangements discussed above raise a number of puzzling issues. First, why should a firm develop additional features for its product if competitors can freely appropriate these features for their products? Second, technology experts have pointed to cases in which COSS products are comparable to or even better than similar products produced

[1] Open source software should be distinguished from two other forms of freely available software. Some firms make their software available for free ("freeware") but do not make the source code available (e.g., Adobe Reader). Another form is voluntary open source, where a firm releases the source code but with strong restrictions on its use and redistribution. We do not consider these cases because the strategic issues involved differ significantly from those COSS firms face.

[2] Popular examples of open source software include the Linux operating system, Firefox browser, OpenOffice, Apache Web Server, SugarCRM, and MySQL, among others. See http://www.sourceforge.net for a Web-based repository of open source applications.

[3] We focus on the two types of licenses most common and relevant to the COSS industry: the GNU General Public LICENSE (http://www.gnu.org/licenses/licenses.html#GPL) and the Berkeley Software Distribution License (http://www.opensource.org/licenses/bsd-license.php). See Laurent (2004) for more discussion.

by traditional closed source software firms (Dedeke 2009). How does a market in which firms face a strong incentive to free ride produce high-quality products? Third, does making sharing mandatory result in the creation of more features? Fourth, when is the "cancer" of free riding, as Ballmer describes it, likely to hurt firms and consumers? Despite the growing importance of the COSS industry, little extant research examines firms' competitive strategies in this novel setting. To address these questions, we incorporate the unique aspects of this industry into a model and use it to analyze firms' competitive strategies to shed light on these empirical puzzles.

We provide a brief overview of the model and then discuss our key results in more detail. Our model has two interacting markets: a *product market* consisting of COSS firms that sell software products to consumers and a *developer market* in which firms hire developers to create software products.

The product market is a vertically differentiated duopoly of ex ante identical software firms that choose product quality and prices. A product's quality is a function of two components: features and usability. We consider two types of product market interaction that are governed by the terms of the open source license. In a private features market, a firm can freely use any open source features and include them in its software product, without restriction. In contrast, under a shared features market, features firms develop must be contributed to open source, so each firm also gets access to the publicly available features the competition develops. A shared features market enables each firm to free ride on the features contributions of other firms. Usability improvements are always kept private in both types of product market; firms do not have access to a competitor's usability components, regardless of the license.

In the developer market, firms hire developers to create more functionality (features) for their products. However, firms do not know whether a particular developer has the appropriate skill level to be hired. Open source provides a mechanism for highly skilled developers to signal their skill (type) to firms by contributing features to open source. Open source contributions provide a credible signal to firms about a developer's skills because potential employers can review a developer's contributions (Leppamaki and Mustonen 2009). Such a motivation to contribute is consistent with an economic signaling rationale to explain developers' open source contributions (Lerner and Tirole 2002).

The interaction between the product and developer markets in equilibrium determines developers' wages, the degree of open source software created by firms and through developers' signaling, as well as product qualities and prices. Developers' wage expectations are influenced by the structure of product market competition discussed above and effect developers' incentives to contribute to open source. A higher expected wage increases open source feature contributions. Higher wages affect firms' decisions to develop more features, which has a feedback effect on the wage offered. Equilibrium product quality balances developers' incentives to contribute to open source and firms' willingness to pay for marginal product improvements. Thus, understanding and modeling the creation of open source software is critical to understanding the competitive product design and pricing strategies of COSS firms.

We demonstrate how the puzzles raised earlier in this section result from the competitive strategies of COSS firms across the product and developer markets, and we compare the models' equilibrium outcomes in the shared features and private features markets. Our key results, mirroring the puzzles, are the following.

First, in the shared features market, we find that free riding on features is supported in equilibrium: the (ex post) high-quality firm creates additional open source features, whereas the low-quality firm does not. Both firms also develop some degree of usability. The high-quality firm contributes to open source because the complementary nature of features and usability increases the value of differentiating on usability, and both firms appropriate the benefits from quality differentiation. The low-quality firm has less incentive to contribute features because it can free ride on the high-quality firm, and its marginal value of additional features is lower, consistent with empirical findings that the high-quality firm Red Hat contributes significantly more code to Linux than competing vendors (Pal and Madanmohan 2002, Handy 2008). Second, diminished competition between firms in the developer market explains higher quality in the shared features market, in conjunction with production efficiencies created by mandatory sharing. Third, open source contributions can be higher in larger private features markets because developers expect higher wages from intense competition between firms for their skills, which increases developers' incentives to contribute to open source features. Fourth, both consumers and the low-quality firm are unambiguously better off in the free-riding shared features market than in the private features market. Under certain conditions, the high-quality firm may also earn higher profits in the shared features market. Consumer surplus is higher with free riding because of increased price competition, resulting from reduced product differentiation through common features.

Although we capture the most salient competitive factors in the COSS marketplace, we must abstract

away from several interesting issues, such as the voluntary provision of open source and multiproduct firms. We focus our study on firms' strategies given a specific COSS market. We discuss these limitations and ideas for future research in §6.

Our paper connects two distinct and previously separate streams of literature, especially in modeling the interconnection between the product and developer markets. The product market model is based on well-known work in marketing and industrial organization on strategic product development with vertically differentiated firms (Shaked and Sutton 1982, Moorthy 1988). The novelty in our setting is the combination of public and private goods that is a unique aspect of COSS products, and thus it contributes to the nascent literature in cocreation of products by firms and by a community of individuals working outside the traditional firm boundaries (Ramaswamy and Gouillart 2010, Sawhney et al. 2005). Our work is also closely related to, but quite distinct from, several recent studies on open source that have examined issues such as two-sided pricing of operating systems and applications (Economides and Katsamakas 2006) and dynamic competition between open and closed source products (Casadesus-Masanell and Ghemawat 2006). Neither paper explores product design or the strategic interaction between open source firms, and our work complements these two papers because we examine COSS products that are based on open source and that compete with each other. Leppamaki and Mustonen (2009) assume a perfect market for developers, ignore the multidimensional aspect of quality, and do not explore strategic interaction between COSS firms. Casadesus-Masanell and Llanes (2011) consider how a firm's decision to release its software as open source depends on whether a competitor uses an open source or proprietary business model. This model corresponds to the case of MySQL, whereas our model fits better with Linux and Red Hat. One critical difference is that the two-sided nature of our market endogenizes the creation of open source. Another is that firms in our model construct software based on existing open source code contributed by developers. Thus, firms are unable to dictate the terms of the license. We leave to future research the possibility that firms could endogenously choose whether to operate in one or more open source markets.

The second stream of literature helps in understanding why developers contribute to open source software. The literature distinguishes between intrinsic motivation (including altruism, use value, and similar factors) and extrinsic motivation (driven by market factors, developer status, and potential wages), and it finds that extrinsic factors play a major role in developers' decisions to contribute (Roberts et al. 2006). Hars and Ou (2002) use direct surveys of developers to provide support for signaling, and Fershtman and Gandal (2007) show that developers contribute more when the license is more commercially oriented. Lerner and Tirole (2002) argue in a review article that much of the evidence is consistent with an economic signaling motivation.[4] Based on this work, we build on Spence (1973) to formulate the developer market as individuals signaling to demonstrate an unobservable skill (or ability). We extend Spence's model to include two types of signal spillovers: developer–firm and firm–firm spillovers. We now provide a broad overview of the open source industry to better understand the context of our study.

2. The Open Source Industry

The open source movement gained prominence in the 1990s as a small community of expert developers who made the source code (or blueprints) to their programs freely available for anyone to use and build on. The growth of open source software has been rapid: market researcher IDC (2009) estimates that the direct revenue in this market will grow to $8.1 billion by 2013, with compound annual growth of 22%. In addition, open source software is increasingly being used as part of commercial software. Large technology firms such as IBM, Sun Microsystems, and Hewlett-Packard have long recognized the importance of open source and have launched multibillion-dollar open source initiatives.[5]

Consumers find that the freely available open source software lacks usability and requires significant expertise to use effectively (Lakhani and von Hippel 2003). COSS firms add value to open source by increasing the functionality (or features) and the usability, which are two distinct dimensions of overall software quality:

1. *Features:* Provide new or enhanced features that extend the basic operations of the software. For example, Sun's StarOffice suite has additional features providing compatibility and support for different document formats than the open source OpenOffice.

2. *Usability and Support Services:* Enhance the user's ability to effectively use the product's available features. Usability enhancements often take the form of nonsoftware services, such as online help, technical assistance, documentation, packaging, and other support services.[6]

[4] For anecdotal evidence, two apt examples can be found at http://www.odesk.com/blog/2008/02/open-source-work-as-a-portfolio/ and http://blogs.techrepublic.com.com/opensource/?p=821 (accessed March 11, 2011).

[5] See *InfoWorld* (2000).

[6] Usability can also include software code or additional programs that make the interface more accessible.

The distinctions between these dimensions of functionality and usability that contribute to software quality have been recognized and quantified by standards organizations such as ISO (Abran et al. 2003, Jung et al. 2004). An intuitive way to think about the difference is that expert users are able to effectively use software with complex functionality and low levels of usability, but the productivity of ordinary users increases with the usability of the software. For example, contrast the ability of expert users of spreadsheet software to use custom macros to accomplish complex tasks versus ordinary users' interactions with the software through the menu-driven interface. The distinction plays a key role in our model formulation, and we also incorporate the fact that creating functionality is considered a highly skilled endeavor. Developers who work on usability, known as usability engineers, do not ordinarily work on open source software unless hired to do so by a firm, which is consistent with the well-known lack of usability in open source projects (Lakhani and von Hippel 2003).

Numerous firms have adopted the COSS model, and COSS products now span a wide range of applications, from productivity suites to business intelligence to customer relationship management.[7] Industry professionals clearly recognize that building on open source is an important business strategy (Goldman and Gabriel 2005, Riehle 2007). A prime example of a COSS firm is Red Hat Inc. The company's commercial version of the freely available Linux open source software is designed to simplify and extend the management and administration of the Linux operating system. Red Hat Linux operates under the GNU General Public License, which implies that Red Hat must make publicly available any feature contributions it makes to Linux but can keep private any usability enhancements. Red Hat makes significant contributions to the Linux kernel, the Linux X Windows System, the GNU Compiler Collection, and others, all of which are made public under Linux's license (Pal and Madanmohan 2002, Handy 2008). Red Hat provides additional services, such as extensive documentation, installation and maintenance, and support programs, to customers who purchase their commercial product.

Thus firms may build on open source, and the terms of the license govern whether COSS firms can keep their features private or must make them publicly available. We address this distinction in our model below.

[7] An exhaustive list of commercial applications based on open source can be found at Wikipedia; http://en.wikipedia.org/wiki/Commercial_open_source_applications (accessed March 11, 2011).

3. Model—Private and Shared Features Markets

We model a duopoly of ex ante identical firms competing in two separate but interconnected markets: the first is a product market in which consumers purchase software produced by the firms, and the second is a developer market in which firms compete for developers.

Figure 1 details the sequence of stages in our model, which includes both private features and shared features markets. Stages 0, 1, 3, and 4 are common to both types of product markets, whereas Stage 2^P is only present in the private features market, and Stages 2_a^S and 2_b^S are only present in the shared features market. In Stage 1, developers choose whether and how much to contribute to open source features to signal their skill levels. In the private features market, firms simultaneously observe developer signals and make wage offers and *all* product development decisions in Stage 2^P. In the shared features market, firms observe developer signals and make wage offers and *only* feature development decisions in Stage 2_a^S, and then they choose how much usability to develop and how many features to copy from their competitor in Stage 2_b^S. In Stage 3, firms set prices after observing product qualities, and in Stage 4, consumers make their purchase decisions after observing all qualities and prices. Table 1 summarizes our model's notation, distinguishing between exogenous model primitives and endogenous outcomes in the model.

3.1. Product Market

Consumers choose to either purchase one unit of software from one of the COSS firms or not purchase a

Figure 1 Sequence of Stages in Model

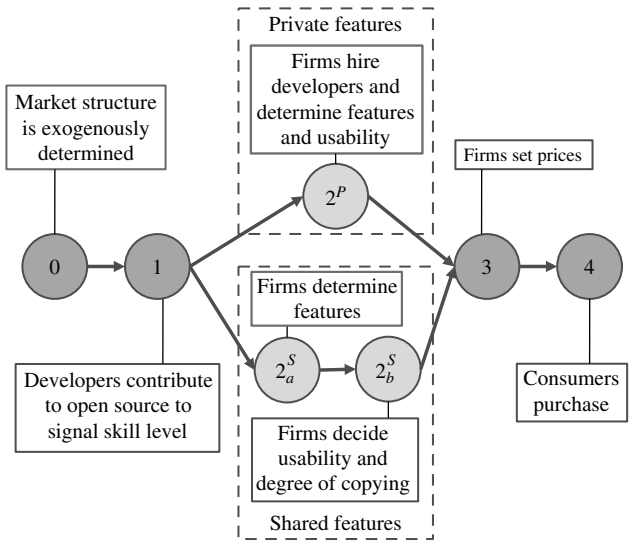


Table 1 Notation for Main Constructs in the Model

Symbol	Definition
Market types	
P, S	Superscripts for private features and shared features markets, respectively
Model primitives	
M	Market size, heterogeneity
c_L, c_H	Cost of contributing one feature to open source for each developer type
c_s	Cost of developing one unit of usability
$\eta_H\ (=\eta), \eta_L$	Productivity of high-type and low-type developers, respectively
Equilibrium strategies and outcomes	
p_j, q_j	Price and quality of firm j's product, respectively
f_j, s_j	Number of developers hired and usability investment by firm j, respectively
F_j	Number of features in firm j's product
F_0	Number of open source features as a result of developer contributions
e_L, e_H	Contribution to open source by low- and high-type developers respectively
$w(e)$	Wage in the developer market, with $e \in \{e_L, e_H\}$

product, in which case they receive a normalized utility of 0. Consumers are heterogeneous in their preferences for quality, and a consumer indexed by θ has utility for a software product of quality q at price p given by

$$u(q;\theta) = \theta q - p.$$

Consumers' marginal valuation for quality is distributed uniformly, $\theta \sim \mathrm{U}[0, M]$, and its realized value determines which product the consumer prefers. M is the market size or heterogeneity parameter, and a higher value indicates a higher dispersion in consumer valuation for quality. Note that the mass of consumers is fixed at 1.

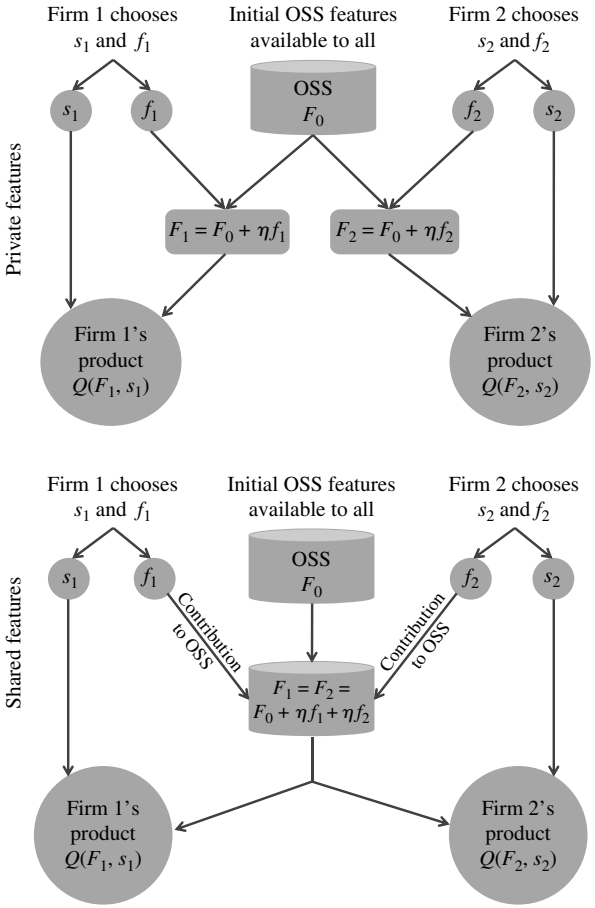
The quality q of a software product depends on its level of features F and usability s. We follow industry practice and view these dimensions as mutually exclusive (Boehm 1981, Pressman 2004). Quality is defined by the production function:

$$q = Q(F, s).$$

A software product's features define the set of tasks the product can help accomplish, whereas usability refers to the ease with which a consumer can make use of the product's features. Consumers value both more features and greater usability. However, an abundance of features may create an overly complex product, and consumers may not be able to take advantage of all the features without usability. Conversely, a high level of usability is more beneficial in conjunction with a large number of features.[8] We

[8] In general, consumers do not benefit from products with a significant imbalance between their level of features and usability. Thompson et al. (2005) show that consumers who purchase overly complex products face "feature fatigue" and that improving usability can help consumers effectively utilize the features.

Figure 2 Product Market: Comparison of Private and Shared Features



Note. OSS, open source software.

therefore model these two dimensions of quality as (imperfect) complements, implying

$$\frac{\partial^2 Q}{\partial s \partial F} > 0.$$

A simple functional form that captures this complementarity and is concave in both features and usability is Cobb–Douglas, $Q(F, s) = (F \cdot s)^{1/4}$, which we use for convenience. The above formulation of consumer preferences and product quality are common to all markets. Figure 2 illustrates the structure of the product market: the upper panel depicts the private features market in which both firms initially have access to F_0 features from the open source community, and the lower panel depicts the shared features market in which firms contribute any features they develop to open source.[9] Firm $j \in \{1, 2\}$ determines its product quality by hiring f_j feature

[9] F_0 is determined in equilibrium through developer signaling (discussed later in this section).

developers and choosing a level s_j of usability, and we denote firm 1 as the firm with overall higher quality, letting $q_1 > q_2$ without loss of generality. Developers vary in their skill level or productivity, $\eta \in \{\eta_L, \eta_H\}$, with a developer producing η units of features when hired. The key difference between the shared features market and the private features market is the formulation for product quality. In the upper panel, firms in the private features market incorporate open source features and only their own privately developed features into their products. In contrast, the shared features market in the lower panel shows that firms must contribute any features they develop back to open source, which could subsequently become part of both firms' products.

In the private features market, firm j's product has $F_j = (F_0 + \eta f_j)$ features. In the shared features market, firms determine how much to copy from their competitor, denoted by $\delta_j \in [0, 1]$ in Stage 2_b^S. The number of features in the product is thus $F_j = (F_0 + \eta f_j + \eta \delta_j f_{-j})$.

In both markets, each firm j also develops the usability of the product to level s_j to make the product's functionality easier for consumers. The overall quality of firm j's product depends on whether features are private (P) or shared (S):

$$q_j^P = Q(F_j, s_j) = [(F_0 + \eta f_j)s_j]^{1/4}, \quad \text{and}$$

$$q_j^S = Q(F_j, s_j) = [(F_0 + \eta f_j + \eta \delta_j f_{-j})s_j]^{1/4}.$$

The production costs in both cases include wages w paid to developers to create features and to enhance usability at a cost of c_s per unit.[10] The total fixed production cost $C(f, s)$ is the sum of feature development costs and usability creation costs and is given as $C(f, s) = w \cdot f + c_s \cdot s$.

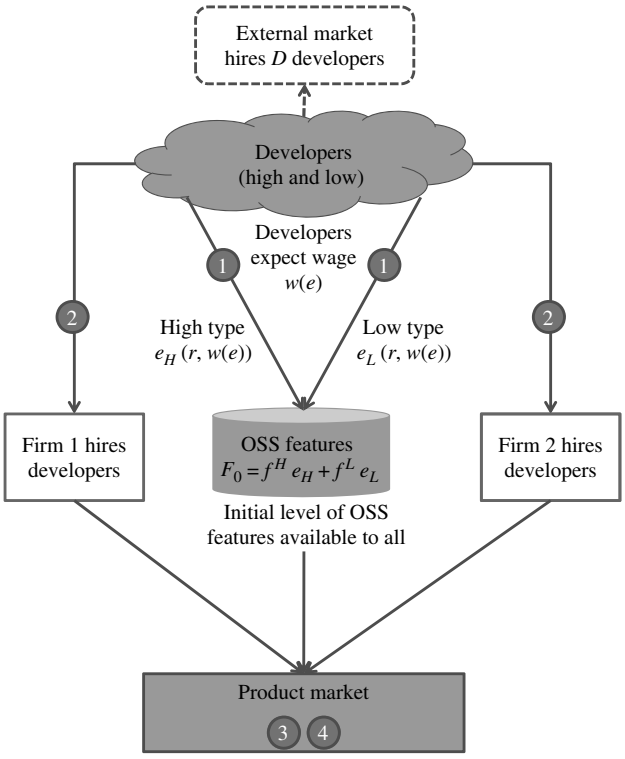
3.2. Developer Market

Developers are heterogeneously distributed and are either highly skilled (high type) or lowly skilled (low type). The cost of contributing to open source differs according to the developer's type. To contribute e features, high types incur a cost of $c_H \cdot e$ and low types incur a cost of $c_L \cdot e$, with $c_H < c_L$ indicating that high types face a lower cost.[11] High-type developers have a reservation option $r \sim \psi(\cdot)$, with cumulative distribution function $\Psi(\cdot)$ and support in the range $[\underline{R}, \bar{R}]$, where $0 \le \underline{R} < \bar{R}$.[12] This option represents the utility a developer derives from her current job, and she will accept a new wage offer only if it exceeds r and the cost of signaling. Note that developers are heterogeneous in two dimensions: their (discrete) skill levels and their (continuous) reservation wages.

After the signal successfully reveals the developer's type or skill level, the developer can choose either of the COSS firms, or an external market. The external market for highly skilled developers is present because skills required to successfully contribute to open source can have useful applications across many markets. For example, a developer who has contributed to a Linux open source project can receive an offer from a firm such as Oracle that is not a primary competitor in the Linux market. We capture by an external demand $D(w)$ the degree of specificity of software skills the developer can signal by contributing to the open source project. A low external demand indicates that the skills signaled are very specific, whereas a high external demand indicates the opposite. Therefore, the external demand, $D(w)$, is a function of the market wage and affects market outcomes only in the sense that it affects the wage by creating an additional source of demand for highly skilled developers. Note that the external market has no demand for developers who have not signaled, because their skill level is uncertain.

Figure 3 displays the developer market and its link to the product market. The numbers correspond to

Figure 3 Developer Market



Note. OSS, open source software.

[10] We focus on the labor market for feature developers because creating new functionality is a more specialized and challenging skill, whereas modeling usability as being exogenous is more realistic.

[11] Although we consider all features to be equivalent for the sake of expositional clarity, our results hold even when developers contribute features that are heterogeneous in "quality." The proof for this case is available from the authors upon request.

[12] No further assumptions on the functional form of Ψ are required for our results. We use the general form in our description because it helps in presenting and interpreting different effects.

the stages in the game outlined in Figure 1. Developers form expectations about wages based on the market and contribute to open source to signal their skill level to firms (Stage 1). These contributions are publicly observable, and firms use them to evaluate developers' skills and make wage offers to developers (Stages 2^P and 2_a^S).[13] Firms simultaneously hire developers, and in doing so, they choose their product's level of features.

Low-type developers may attempt to masquerade as high-type developers to convince firms to hire them. Open source projects have gatekeepers who screen for low-quality code and decide which submissions to accept, implying that high-type developers have a lower cost of contribution (Bagozzi and Dholakia 2006). To simplify our exposition, we make two assumptions: low-type developers have reservation option $R_L \geq 0$, and they are not productive to COSS firms when hired; i.e., $\eta_L = 0$. These assumptions make hiring the low types unprofitable.[14]

If firms could observe a developer's skill level, they would base the wage on this observation. However, when the wage is unobservable, it is dependent on the contribution to open source e, denoted as $w(e)$. A developer of type $t \in \{L, H\}$ contributing e features receives utility $u_t(w, e) = w(e) - c_t e$, and her optimal contribution level is

$$e_t = \arg\max_e w(e) - c_t e, \tag{1}$$

subject to the incentive compatibility (IC) and individual rationality (IR) conditions:

$$\begin{aligned} &(\mathrm{IC}_H)\colon\ w(e_H) - c_H e_H \geq w(e_L) - c_H e_L,\\ &(\mathrm{IR}_H)\colon\ w(e_H) - c_H e_H \geq r,\\ &(\mathrm{IC}_L)\colon\ w(e_L) - c_L e_L \geq w(e_H) - c_L e_H,\\ &(\mathrm{IR}_L)\colon\ w(e_L) - c_L e_L \geq 0. \end{aligned} \tag{2}$$

To ensure separation, high-type developers must find exerting a high level of effort (IC_H) to be incentive-compatible and must be sufficiently compensated to work for the COSS firms compared with her reservation option (IR_H). Low-type developers must find imitating the high-type developers (IC_L) to be prohibitively expensive. The IR constraint for the low type is trivially satisfied. Firms do not find hiring low-type developers at any positive wage to be optimal after their type is credibly revealed. The condition in IC_L reduces to $0 \geq w(e_H) - c_L e_H$ when low-type developers do not contribute. Among high-type developers, those with reservation values below a threshold, $r \leq w(e_H) - c_H e_H$, choose to signal, whereas others with higher reservation options, $r > w(e_H) - c_H e_H$, do not signal. Thus the number of high-type developers who signal is $f_0^H = \Psi(w(e_H) - c_H e_H)$.

This condition characterizes the supply of developers available for hire either by the firms or by the external market. Given an arbitrary wage w, the initial level of open source features available in Stage 1, $F_0(w)$, is a function of the number of developers and their contributions:

$$F_0(w) = f_0 e_H = \Psi(w(e_H) - c_H e_H) e_H. \tag{3}$$

Although the IR and IC conditions are necessary, they are not sufficient to determine wage, which depends on the interaction between the product market and developer market, which govern the demand and supply of developers, respectively.

3.3. Discussion and Comparison

We compare the current setting with the canonical model of Spence (1973), in which workers signal through investments in education to a market of perfectly competitive firms and capture all surplus. In our model, developer contributions to open source features serve as signals to COSS firms seeking to hire highly skilled developers. We build on Spence in two primary areas, thereby making the model applicable to a broader range of settings. First, our model considers an imperfectly competitive, vertically differentiated duopoly in which firms earn positive profits—developer heterogeneity *within* types and different marginal productivity of developers imply that firms only make wage offers based on the marginal value of the last feature to them. Thus, the diminishing marginal valuation of consumers for quality allows both firms and developers to earn a surplus.

Second, the signaling contributions have two types of spillover effects: developer-to-firm spillovers and firm-to-firm spillovers. In developer-to-firm spillovers, contributions developers make in Stage 1 serve as substitutes for firms' feature contributions. The more developers contribute to open source as a result of signaling, the less intensely firms compete for them, which acts to diminish the wage. In the shared features market, firm-to-firm spillovers occur because the feature contributions from each firm are available to the competitor. As a result, firms compete less intensely for developer talent because the firms realize that they can fully appropriate the features developed by their competitor.

In contrast, Spence's (1973) model of education signaling does not consider either imperfect competition or spillover effects of either kind that we have conceptualized. In our model, signaling by a developer not

[13] Firms' wage offers are consistent with developers' beliefs in equilibrium.

[14] We only require that at the equilibrium wage, the marginal increase in a firm's profits from hiring an incremental low-type developer is sufficiently low. This condition would be met if $\eta_L > 0$ but still low enough for firms to find that not offering them a positive wage is profitable.

only affects that developer's utility and the COSS firm that hires him, but it also critically changes the strategic interaction between the COSS firms in the market, and we believe the current paper is the first that examines this important aspect of including strategic spillovers in signaling settings. These conflicting incentives create a critical and unique link between the product and developer markets, and we proceed next to analyze the properties of the interlinked equilibrium in these markets.

4. Equilibrium Analysis

We analyze the subgame-perfect equilibrium of the model by backward induction, beginning with the choices consumers make and the pricing subgame in Stage 3. We then move to the product development decisions in Stages 2^P or $2_a^S/2_b^S$ and finally to the developer signaling game in Stage 1. Intermediate results that are not central to our analysis appear as lemmas in the electronic companion (available as part of the online version that can be found at http://mktsci.pubs.informs.org/).

4.1. Pricing Equilibrium

The pricing subgame in Stage 3 applies to all cases of the product market, and it borrows significantly from Moorthy (1988). We define consumer θ_{12} as indifferent between firm 1's and firm 2's products, so that all consumers with higher value for quality, i.e., $\theta > \theta_{12}$, would prefer firm 1's product. θ_{12} is determined by the indifference condition, $\theta_{12} \cdot q_1 - p_1 = \theta_{12} \cdot q_2 - p_2$. Similarly, we define consumer θ_{20}, who is indifferent between purchasing firm 2's product and not purchasing any product, such that $\theta_{20} q_2 - p_2 = 0$. Note that all consumers θ such that $\theta_{20} < \theta < \theta_{12}$ choose to purchase firm 2's product over firm 1's product or the no-purchase option. The market shares for the firms are then determined to be

$$m_1 = \frac{1}{M}(M - \theta_{12}) = \frac{1}{M}\left(M - \frac{p_1 - p_2}{q_1 - q_2}\right), \quad \text{and}$$

$$m_2 = \frac{1}{M}(\theta_{12} - \theta_{20}) = \frac{1}{M}\left(\frac{p_1 - p_2}{q_1 - q_2} - \frac{p_2}{q_2}\right).$$

Revenues are determined from the above market shares and prices, and firms set prices to maximize revenues, leading to the optimal price levels as a function of the quality levels of both firms' products:

$$p_1^* = \arg\max_{p_1}\left(M - \frac{p_1 - p_2}{q_1 - q_2}\right)p_1, \quad \text{and}$$

$$p_2^* = \arg\max_{p_2}\left(\frac{p_1 - p_2}{q_1 - q_2} - \frac{p_2}{q_2}\right)p_2.$$

The optimal price for each firm is a best response of the price the other firm sets and depends on the quality levels of both products, and the optimal revenues are a function only of product quality levels chosen by the firms.

At this stage, we note that firm revenues, prices, and the consumer surplus are represented solely as functions of product quality levels and the market size. Next we characterize the product design choices concerning features and usability, and we evaluate how that affects product quality.

4.2. Product Quality Equilibrium

Firms determine their product features by hiring developers and making usability investments, both of which contribute to product quality. The product strategy subgame begins with Stage 2^P for the private features market and at Stage 2_a^S for the shared features market. We characterize the partial-equilibrium behavior by the COSS firms given the wage (w) and freely available open source features (F_0). The profit functions are derived by substituting the feature and usability levels into the revenue functions and accounting for the costs of development.

The solution of the subgame in Stage 2^P for the private market and in Stages 2_a^S and 2_b^S for the shared features market requires the firms to strategically determine the optimal mix of features and usability, which in turn determines overall quality. In the shared features market, the potential for free riding on features leads to decreased product differentiation and increased price competition, making it unclear whether any firm will contribute to features in equilibrium or whether each firm will find free riding on the contributions its competitor makes to be more profitable.

Proposition 1 (Product Market Equilibrium). *The product strategy equilibrium resulting from F_0 initial open source features, and with firms $j \in \{1, 2\}$ differentiated on the basis of product quality, is characterized by the level of features, usability, and overall quality, depending on the market type*:

Private	*Shared*
$s_j^P = \pi_j^S M^2 \sqrt{\dfrac{\eta}{{c_s}^3 w}}$,	$s_j^S = \psi_j^Q M^2 \sqrt{\dfrac{\eta}{{c_s}^3 w}}$,
$f_j^P = \pi_j^F M^2 \sqrt{\dfrac{\eta}{c_s w^3}} - \dfrac{F_0}{\eta}$,	$f_1^S = \psi_1^F M^2 \sqrt{\dfrac{\eta}{c_s w^3}} - \dfrac{F_0}{\eta}$,
	$f_2^S = 0$,
$q_j^P = \pi_j^Q M \sqrt{\dfrac{\eta}{c_s w}}$,	$q_j^S = \psi_j^Q M \sqrt{\dfrac{\eta}{c_s w}}$.

In the shared features market, each firm fully copies its competitor's features, such that $\delta_1 = \delta_2 = 1$.

The constants π^S, π^F, and π^Q for the private features market and ψ^S, ψ^F, and ψ^Q for the shared features market are defined in the electronic companion.

There are several points to note here. First, in the private features market, both firms create features in addition to the F_0 features available from open source. An increase in F_0 leads firms to reduce the number of features they develop because the open source features are a substitute. We find that firms differentiate their products more on the less expensive dimension of quality, implying that if features are less expensive to produce, the firms will differentiate more on features ($f_1 - f_2 > s_1 - s_2$). Intuitively, firms differentiate their products more on the dimension that yields a greater return to such differentiation.

Second, the implication in Stage 2_b^S of the shared features market is that both firms fully copy their competitor's features. A partial copying strategy might seem optimal because it would create more product differentiation. To understand the intuition behind this result, consider each firm's incentives separately. For the high-quality firm, copying more is always preferred because doing so increases the firm's own quality and the degree of quality differentiation ($q_1 - q_2$) between the firms. For the low-quality firm, copying increases its own quality but reduces quality differentiation. However, the positive effect of an increase in quality dominates the negative effect of a reduction in differentiation when the quality levels are sufficiently far apart. The low-quality firm copies features fully from the high-quality firm and, consequently, invests less in usability to attain a certain quality level.

We find that only the high-quality firm develops features beyond the open source features F_0 available as a result of developer contributions. The low-quality firm completely free rides on the features the high-quality firm provides. Why does the high-quality firm develop a positive level of features? The intuition comes from the fact that the quality dimensions are complements, implying that consumers' marginal utility of usability is increasing in the level of features. Although features do not contribute directly to product quality differentiation, they do magnify the effect of usability differences between the firms. Note that the low-quality firm invests less in usability than the high-quality firm, and the high-quality firm develops features to enhance the degree of product differentiation, which results from having a higher usability product. Thus the increase in the differentiation from usability makes creating features that reduce the intensity of competition worthwhile.

The product market outcome in the shared features market above has a critical implication for the developers' market, which we examine in the next section: only the high-type firm hires developers, because the low-type firm does not develop features ($f_2 = 0$), and only the external market offers an alternative to developers who signal.

4.3. Developer Market Equilibrium

The wage influences firms' demand for developers, the number of developers willing to signal, and the level of their contributions to open source in Stage 1. The equilibrium wage must balance firms' demand for developers with the supply of developers who are willing to enter the market by signaling. We focus on separating equilibria where high-type and low-type developers make different contributions to open source software.

For a separating equilibrium in the signaling game, we identify conditions on wages that lead to positive contributions by some high-type developers and that ensure low-type developers do not find imitating them to be profitable. The following result provides the necessary conditions for a least-cost separating (LCS) equilibria in terms of the wage and open source contributions. The contributions and beliefs that correspond to the least-cost separating equilibrium, including the out-of-equilibrium beliefs that satisfy the intuitive criterion (Cho and Kreps 1987), help determine a unique equilibrium, to which we restrict further attention.[15]

Proposition 2 (Developer Market Separating Equilibrium). *In the LCS equilibrium, both the number of high-type developers contributing to open source and the number of features contributed by each developer increase with the wage.*

Note that this result is independent of the type of product market. The contribution decisions of both types of developers determine the minimum amount of contribution required for separation; $e^{\mathrm{LCS}}(w) = w/c_L$. The higher the wage, the more incentive there is for the low-type developers to masquerade as high-type developers, and therefore the latter must contribute even more features to make it unattractive for the former to masquerade. Highly skilled developers with low reservation or outside options, i.e., $r < r^{\mathrm{LCS}}(w) = w - c_H e^{\mathrm{LCS}}(w) = w(1 - c_H/c_L)$, choose to signal, implying that the number of high-type developers who signal is $f_0^{\mathrm{LCS}} = \Psi(r^{\mathrm{LCS}}) = \Psi(w - c_H e^{\mathrm{LCS}}(w))$. Thus, as the wage increases, the number of high-type developers who will find contributing to open source relatively more attractive compared to their current reservation option increases as well.

The number of open source features produced in Stage 1 is the product of the number of developers who signal and each developer's contribution,

[15] Signaling models often admit a multiplicity of equilibria, and we use the intuitive criterion to refine "unreasonable" equilibria. In our context, least cost refers to the minimum separation required at each prevailing wage. This purification of out-of-equilibrium beliefs requires that any observed deviation from the equilibrium path will more likely be from the type that could profit the most from the deviation.

$F_0^{\mathrm{LCS}}(w) = f_0^{\mathrm{LCS}} e^{\mathrm{LCS}}(w) = (w/c_L)\Psi(w(1 - c_H/c_L))$. This result allows us to focus on the (expected) wage level that determines F_0. We restrict further attention to LCS equilibrium outcomes and drop the LCS superscript.

5. Market Interaction and Comparison

In the previous results, we separately examined the product market competition involving firms and consumers as well as the developer market involving firms and developers. We integrate these two sides of the overall market to understand competitive product strategy across private features and shared features markets. The key connection between the markets is the wage for developers, which determines how many features are produced for inclusion in the products.

The equilibrium wage w^P or w^S equates the overall demand for developers, formed by the firms' demand and the external demand $D(w)$ for developers, with the aggregate supply of developers who prefer signaling to their reservation option. We denote the excess demand functions ξ^P for the private features market and ξ^S for the shared features market for developers at wage w to be

$$\begin{aligned}\xi^P(w) &= D(w) + M^2\sqrt{\frac{\eta}{c_s w^3}}D(\pi_1^F + \pi_2^F) \\ &\quad - 2\frac{w}{c_L}\Psi\left(w\left[1 - \frac{c_H}{c_L}\right]\right) - \Psi\left(w\left[1 - \frac{c_H}{c_L}\right]\right), \\ \xi^S(w) &= \underbrace{D(w)}_{\text{External demand}} \\ &\quad \underbrace{+ M^2\sqrt{\frac{\eta}{c_s w^3}}\psi_1^F - \frac{w}{c_L}\Psi\left(w\left[1 - \frac{c_H}{c_L}\right]\right)}_{\text{COSS demand } = f_1 + f_2} \\ &\quad \underbrace{- \Psi\left(w\left[1 - \frac{c_H}{c_L}\right]\right)}_{\text{Developers' signaling}}. \end{aligned} \tag{4}$$

Equating the excess demand to 0 determines the equilibrium wage in either case. Below this wage level, fewer high-type developers will contribute to open source than firms are willing to hire, and above this wage level, more developers will be inclined to contribute than firms will want to hire. Thus the equilibrium wage serves to balance the signaling incentives of developers and firms, and we examine its comparative statics with respect to the market size, the cost of signaling, and the cost of producing usability. The implicit equations $\xi^P(w^P) = 0$ or $\xi^S(w^S) = 0$ for private features and shared features markets, respectively, determine the equilibrium wage w^P or w^S.

Proposition 3 (Wages). *The following properties hold for the equilibrium wage.*

(i) *The wage in the shared features market is lower than in the private features market,* $w^S < w^P$.

(ii) *In both markets,* w *satisfies the following properties:* w *is increasing in market size* $(\partial w/\partial M > 0)$, *signaling cost* $(\partial w/\partial c_H > 0)$, *and skill level* $(\partial w/\partial \eta > 0)$; *and it is decreasing in the cost of usability* $(\partial w/\partial c_s < 0)$.

In (i), we find that the equilibrium wage is lower in the shared features market because of free riding between the firms. We find from Proposition 1 that $f_2 = 0$ in the shared features market independent of the wage paid to developers and that firm 2 chooses to completely free ride and copy firm 1's features. The effect of free riding reduces competition between the firms in the developer market, leading to lower developer wages, which in turn leads to fewer open source features produced in Stage 1.

Examining the comparative statics in (ii) we find that the wage increases with the market size M because a larger market causes firms to invest more in creating a higher-quality product, and competition between firms drives wages higher. The wage is higher when producing usability is less expensive, because a low c_s permits firms to invest more in usability. A higher level of usability raises the value of complementary features for consumers, and firms invest more in features, thereby raising developer wages. When signaling is costly for the high-type developers, fewer signal and more choose their reservation options, resulting in fewer open source features, and this diminished supply raises wages.

We next examine the provision of open source features, which is of interest to expert users as well as policy makers and initial creators of open source projects, who may seek to maximize the amount of publicly available features.

Proposition 4 (Creation of Open Source). *The creation of open source features is higher under the private features market than the shared features market when the market size is large and signaling costs for high-type developers are low.*

In the shared features market, both firms have access not only to the contributions to open source high-type developers make but also to the features the high-quality firm develops. In contrast, the private features market allows firms to keep their features private, leaving only the signaling contributions to features as open source, leading us to expect fewer open source features. However, when the consumer market is large, producing a higher-quality product is of greater value, and firms compete for developers more intensely in the private features market. This effect raises the incentive for high-type developers to separate themselves from low-type developers. When separation is relatively easy (c_H is small compared

with c_L), more high-type developers enter the market and more developers contribute to open source features.

The level of open source features, along with firm decisions, impacts overall product quality, to which we turn our attention.

Proposition 5 (Product Quality). *Comparing the markets we find the following.*

(i) *The private features market provides the lowest quality level.*

(ii) *For small market size and low signaling costs, the private features market provides the highest quality product.*

(iii) *The usability ratio s_1/s_2 is always larger for the shared features market than for the private features market, and the features ratio f_1/f_2 is larger under the private features market. The quality differentiation ratio q_1/q_2 captures is higher for the private features market than for the shared features market.*

The low-quality firm's quality is always higher in the shared features market because the firm is able to free ride on the features the high-quality firm provides. This effect holds even when the low-quality firm develops a lower level of usability than in the private features market, and it is independent of the model parameters, such as market size or signaling costs.

The effect of the product market type on the high-quality firm's quality level is more nuanced: it depends on the market size, signaling cost, and cost to develop usability. If the wages were the same, the private features market would offer the highest quality product. Recall, however, that $w^P > w^S$, and the difference between the wages is closest when competition between firms for developers is not as intense. Lower competition for developers could result from a smaller market opportunity. When the market size is large, demand for developers rises, and the cost of developing a higher-quality product and the equilibrium wage increases. These wage effects are stronger for the private features market than for the shared features market, and the quality level the high-quality firm chooses can become higher in the shared features market. When the signaling cost is higher for the high-type developers, this also leads to increased competition that amplifies the competition between the firms for developers in the private features market.

The marginal benefit of usability increases with the level of public contribution of features, which implies that more signaling by developers increases the firms' incentives to develop usability. In the shared features market, firms differentiate more on usability because it is the only means of differentiation, but this differentiation is insufficient to overcome the fact that both products have the same level of features, and the quality differentiation in the shared features market is therefore lower.

Next, we evaluate the level of surplus consumers and firms obtain in both markets.

Proposition 6 (Profits and Consumer Surplus). *Examining the creation and distribution of surplus across firms and consumers, we find that*

(i) *The high-quality firm can make a higher profit under the shared features market.*

(ii) *Consumer surplus is higher in the shared features market than in the private features market under all market conditions.*

The high-quality firm may make a higher profit because a larger external demand induces a broad base of developers contributing to open source, which increases the developer–firm spillover but reduces the firm–firm spillover (free riding on features). In such a situation, the high-quality firm can do better under a shared features market.

The consumer surplus result in (ii) is surprisingly general and counterintuitive: a reduction in firm competition for developers *increases* surplus to consumers. The reasoning is that both the utility of the signal (open source features) used in the products the firms develop and decreased competition in the developers market as a result of free riding on features by the low-quality firm serve to benefit consumers. Thus, contrary to Ballmer's notion that a mandatory sharing license is a "cancer that attaches itself," we find such a requirement may benefit both consumers and firms.

6. Conclusion, Limitations, and Further Work

The commercial open source software market is growing rapidly and has many strategic ramifications for the regular software market (*Economist* 2009). The ability of firms to leverage open source alters product design decisions more than the standard settings in which a product's components are entirely proprietary. We construct a two-sided model of the commercial open source market in which firms compete in both product and developer markets. Developers contribute to open source to signal their skill level to firms, which determines the level of open source available to firms for their products. Firms hire developers to build their products, and then they compete in a vertically differentiated market for consumers. We compare equilibrium outcomes under two common types of open source licenses. We believe our framework can help marketing scholars understand an area with significant conceptual differences that have a strong impact on competitive marketing strategies and that determine the amount of the public

good created in the first place. Future work could explore a number of areas, such as a firm's choice of which license to adopt and whether to operate in multiple open source markets at the same time.

Our model helps rationalize several puzzles observed in the industry, such as why Red Hat contributes significantly more to Linux than any other firm and why a market with mandatory sharing can actually produce higher-quality products than a proprietary market. We show that the mandatory sharing of feature contributions between firms can actually raise profits and consumer surplus and that firms in a market that enables free riding may produce higher-quality products, compared with a market in which firms may privately appropriate investments. Our model also extends prior work on modeling signaling on the job market (Spence 1973) by including two types of spillovers: developer–firm and firm–firm, and we demonstrate how firms and consumers benefit from these spillovers.

As usual, conducting such an analysis in the context of a formal model entails making numerous simplifying assumptions and abstractions, which we discuss here, and we evaluate several dimensions along which the current paper may be extended.

First, the market structures we examine in this paper are not an exhaustive list of all the types of markets associated with open source, including cases in which firms develop proprietary software and voluntarily release the source code—for example, MySQL, a database product that released its source code and permits users to modify it for noncommercial purposes but requires a negotiated licensing agreement for commercial applications.

Second, firms face a variety of "market entry"-type decisions, including whether to enter specific markets, which is likely to depend on firm resources, experience, and capabilities our model does not capture but that would be interesting aspects for further study. A firm may face the choice of entering one of many open source markets (e.g., private features versus shared features markets); which market should it enter? If these markets are distinct and the firm ignores its competitor's entry decisions, then the model in this paper would apply separately to each market setting.

Third, the incentives for firms to contribute to open source are more complicated in multiproduct or forward-looking settings. One reason a multiproduct firm may contribute to open source is that such contributions could weaken a competitor in another market. This theory implies that one motivation behind IBM's contributions to Linux, or Sun's contribution to OpenOffice, could be a desire to reduce Microsoft's market dominance (and revenues) and to enable each firm to compete more effectively.

Fourth, we abstract away from any horizontal differentiation in open source products. Firms may develop software that is not general-purpose but instead targeted at a specific niche of users. For example, some versions of Linux target the embedded or mobile device market, where the focus may be on low-power computation rather than access to high-power functionality.

In summary, the market for software built on open source is growing rapidly, and open source has recently made the leap to mobile computing platforms with the release of the Google Android operating system. In the mobile market, hardware manufacturers produce smartphones based on Android and release the source code in a setting that closely resembles the shared features market in our model. Our setting naturally extends to evaluating and understanding the competition between manufacturers in the handset market. Overall, we expect the open source market to attract significant attention from researchers in the future who want to examine the unique aspects of product design, pricing, and firm strategy in this industry.

7. Electronic Companion

An electronic companion to this paper is available as part of the online version that can be found at http://mktsci.pubs.informs.org/.

Acknowledgments

The authors thank Anthony Dukes, Oded Koenigsberg, seminar participants at Stanford University, the University of Chicago, the University of Michigan, the University of Texas at Dallas, the 3GTM Conference at HEC Montréal, and the Marketing Research Forum at Cheung Kong Graduate School of Business for comments. Any remaining errors are their own.

References

Abran, A., A. Khelifi, W. Suryn, A. Seffah. 2003. Usability meanings and interpretations in ISO standards. *Software Quality J.* **11**(4) 325–338.

Bagozzi, R. P., U. M. Dholakia. 2006. Open source software user communities: A study of participation in Linux user groups. *Management Sci.* **52**(7) 1099–1115.

Boehm, B. W. 1981. *Software Engineering Economics*. Prentice-Hall, Englewood Cliffs, NJ.

Casadesus-Masanell, R., P. Ghemawat. 2006. Dynamic mixed duopoly: A model motivated by Linux vs. Windows. *Management Sci.* **52**(7) 1072–1084.

Casadesus-Masanell, R., G. Llanes. 2011. Mixed source. *Management Sci.* **57**(7) 1212–1230.

Cho, I. K., D. M. Kreps. 1987. Signaling games and stable equilibria. *Quart. J. Econom.* **102**(2) 179–221.

Dedeke, A. 2009. Is Linux better than Windows software? *IEEE Software* **26**(3) 103–104.

Economides, N., E. Katsamakas. 2006. Two-sided competition of proprietary vs. open source technology platforms and the implications for the software industry. *Management Sci.* **52**(7) 1057–1071.

Economist, The. 2009. Open-source software in the recession: Born free. (May 28) http://www.economist.com/node/13743278.

Fershtman, C., N. Gandal. 2007. Open source software: Motivation and restrictive licensing. *Internat. Econom. Econom. Policy* **4**(2) 209–225.

Fuller, T. 2003. How Microsoft warded off rival. *New York Times* (May 15) http://www.nytimes.com/2003/05/15/technology/15SOFT.html.

Gartner Group. 2008. Gartner highlights key predictions for It organisations and users in 2008 and beyond. Press release (January 31), Egham, UK. http://www.gartner.com/it/page.jsp?id=593207.

Goldman, R., R. P. Gabriel. 2005. *Innovation Happens Elsewhere: Open Source as Business Strategy*, 1st ed. Morgan Kaufmann, San Francisco.

Handy, A. 2008. Red Hat tops list of corporate Linux code contributors. *Software Development Times* (September 18) http://www.sdtimes.com/link/32870.

Hars, A., S. Ou. 2002. Working for free? Motivations for participating in open-source projects. *Internat. J. Electronic Commerce* **6**(3) 25–39.

IDC. 2009. Worldwide open source services 2009–2013 forecast. Report, IDC, Framingham, MA.

InfoWorld. 2000. Open-source platforms: IBM invests almost $1 billion in Linux. (December 18) 16.

Jung, H. W., S. G. Kim, C. S. Chung. 2004. Measuring software product quality: A survey of ISO/IEC 9126. *IEEE Software* **21**(5) 88–92.

Lakhani, K. R., E. von Hippel. 2003. How open source software works: "Free" user-to-user assistance. *Res. Policy* **32**(6) 923–943.

Laurent, L. S. 2004. *Understand Open Source and Free Software Licencing*. O'Reilly, Cambridge, MA.

Leppamaki, M., M. Mustonen. 2009. Skill signalling with product market externality. *Econom. J.* **119**(539) 1130–1142.

Lerner, J., J. Tirole. 2002. Some simple economics of open source. *J. Indust. Econom.* **50**(2) 197–234.

Moorthy, K. S. 1988. Product and price competition in a duopoly. *Marketing Sci.* **7**(2) 141–168.

Pal, N., T. R. Madanmohan. 2002. Competing on open source: Strategies and practise. Working paper, Massachusetts Institute of Technology, Cambridge. http://opensource.mit.edu/online_papers.php.

Pressman, R. B. 2004. *Software Engineering: A Practitioner's Approach*. McGraw-Hill, New York.

Ramaswamy, V., F. Gouillart. 2010. *The Power of Co-Creation: Build It with Them to Boost Growth, Productivity, and Profits*. Free Press, New York.

Riehle, D. 2007. The economic motivation of open source software: Stakeholder perspectives. *Computer* **40**(4) 25–32.

Roberts, J. A., I. Hann, S. A. Slaughter. 2006. Understanding the motivations, participation, and performance of open source software developers: A longitudinal study of the Apache projects. *Management Sci.* **52**(7) 984–999.

Sawhney, M., G. Verona, E. Prandelli. 2005. Collaborating to create: The Internet as a platform for customer engagement in product innovation. *J. Interactive Marketing* **19**(4) 1–15.

Shaked, A., J. Sutton. 1982. Relaxing price competition through product differentiation. *Rev. Econom. Stud.* **49**(1) 3–13.

Spence, M. 1973. Job market signaling. *Quart. J. Econom.* **87**(3) 355–374.

Thompson, D. V., R. W. Hamilton, R. T. Rust. 2005. Feature fatigue: When product capabilities become too much of a good thing. *J. Marketing Res.* **42**(4) 431–442.