PRODUCTION AND OPERATIONS MANAGEMENT

POMS

Vol. 17, No. 1, January-February 2008, pp. 29–43
ISSN 1059-1478 | EISSN 1937-5956 | 08 | 1701 | 0029

DOI 10.3401/poms.1070.0004

Open Source Development with a Commercial Complementary Product or Service

Ernan Haruvy, Suresh P. Sethi
School of Management, University of Texas at Dallas, Richardson, Texas 75083
{eharuvy@utdallas.edu, sethi@utdallas.edu}

Jing Zhou
Belk College of Business, University of North Carolina at Charlotte, Charlotte, North Carolina 28223,
jzhou7@uncc.edu

We examine optimal control decisions regarding pricing, network size, and hiring strategy in the context of open source software development. Opening the source code to a software product often implies that consumers would not pay for the software product itself. However, revenues may be generated from complementary products. A software firm may be willing to open the source code to its software if it stands to build a network for its complementary products. The rapid network growth is doubly crucial in open source development, in which the users of the firm's products are also contributors of code that translates to future quality improvements. To determine whether or not to open the source, a software firm must jointly optimize prices for its various products while simultaneously managing its product quality, network size, and employment strategy. Whether or not potential gains in product quality, network size, and labor savings are sufficient to justify opening the source code depends on product and demand characteristics of both the software and the complementary product, as well as on the cost and productivity of in-house developers relative to open source contributors. This paper investigates these crucial elements to allow firms to reach the optimal decision in choosing between the open and closed source models.

Key words: pricing research; optimal control; open source; network externalities
History: Received: August 2004; Revised: March 2005, August 2005, and December 2005; Accepted: January 2006.

1. Introduction

The open source development paradigm has proven very successful in recent years in meeting a wide variety of software needs. Open source projects such as Linux, Apache, and Perl have become enormously successful, trampling bigger and better endowed closed source competitors. Their well-publicized success has inspired tens of thousands of open source programmers worldwide and has given a boost to thousands of smaller open source projects (Schiff 2002). In open source development, volunteers—most often users of the software—collaborate on the development of the software by contributing code and bug reports and by sharing and disseminating the code. Advantages resulting from this form of development include savings on labor costs, increased creativity associated with freedom from company restrictions, and greater development speed arising from rapid dissemination and higher relevance to end user needs (Raymond 2001). The major caveat is that the code is freely distributed, making it difficult to sell that very same code. Given this characteristic of open source development, a firm wishing to pursue this development model must first carefully formulate its revenue model for the project and weigh the productivity benefits and labor savings against the loss of potential revenue for its software product.

The decision to embrace the open source paradigm and the formulation of related strategies such as price, quality, and hiring critically depend on the business model that is used by the firm to generate revenues for its products. Given that the open source code is free, a firm that coordinates and invests its own resources in an open source project generally has a commercially sold product that would benefit from the open source code—either by nesting the code within its own commercially sold product (this strategy would be consistent with the two-layer design, e.g. Cusumano and Selby 1995) or by exploiting the complementarities that are present between the code and one of its commercially sold products. Although we cannot rule out altruism and social considerations on the part of firms (e.g., Lee and Mendelson 2007)—and many firms claim to be motivated by such considerations—we feel it is safe to assume a profit motive in for-profit firms. The public relations aspect of open source sponsorship may be another motive,

and the modeling of such benefit is outside the scope of the present work.

Haruvy et al. (2003) evaluate a model in which the open source code is sold as part of a commercial product. In that scenario, open source code may become a component of closed source code and the software can be made commercial. Making open source a component of a commercial product can be done while keeping the source open by providing packaging, distribution, service, or brand name; or by closing the source, such as Microsoft's use of the BSD code or the BSDi operating system (derived from Unix BSD).

In this work we examine a model in a monopoly setting where the open source code is free but complements another product that is sold commercially. The importance of complementarity in the software and technology markets with network externalities has been discussed in the economics literature (e.g., Katz and Shapiro 1985, Shapiro and Varian 1999, Parker and Van Alstyne 2005). In recent years, the idea of complementarity has received particular attention in the area of platform competition in two-sided markets, where platform owners are separate from the application developers (Rochet and Tirole 2003, Armstrong 2006, Armstrong and Wright 2007). Economides and Katsamakas (2006) extend this idea to open source and show that application developers will wish to promote open source. Our work is consistent with these ideas, but has three differences. First, although a subset of platform problems (not two-sided) can be handled with the present framework, our work is not restricted to platforms. Second, if one wanted to use the present framework in a platform setting, the ownership structure of the platform and the application cannot be separate in our model. Last, open source is shown to be more than just a free software or platform. We show that a closed source can serve as freeware; indeed, our analysis finds that a firm in many instances will keep the source closed yet provide it at no charge to generate faster dissemination of its commercial version. The distinction is that open source is an active tool for improving the quality of the software or platform.

The open source business models discussed in the literature (Raymond 1999, 2001; Schiff 2002) suggest several variations on the concept of a free open source product to promote a complementary product: the Market Positioner model uses the open source code to establish brand name and help the firm's other products. Netscape's Mozilla Web browser is an example of such a practice. Netscape's early business model relied upon the sale of server software. The decision to open the source for Mozilla in early 1998 was possibly based on fears that Microsoft would monopolize the browser market and would eventually drive Netscape out of the server software business (see Raymond 1999). Another example is Sun's Star Office which complements Sun's product lines and brand image. The Compatible Hardware model has an open source software compatible with the firm's commercially sold hardware. An example is the Apple Mac OS X software, which is compatible with Apple's hardware. This software has been open source since Apple Computer's decision in mid-March 1999 to open source "Darwin," the core of its Mac OS X server operating system (for more details, see http://www.opendarwin.org). The Service and Support model, of which Red Hat is a prominent example, provides service and support for a fee to complement the open source code. Finally, under the Information model, a firm charges for information associated with the open source code. Consulting services often charge for information relating to open source documentation, maintenance, etc. Notice that the models discussed here can also apply to freeware—software products that are free but whose code is not necessarily open to the public. However, the key difference lies in the development efforts, which are less expensive and more rapid in the open source model. For a discussion of when the freeware model would be most appropriate, see Haruvy and Prasad (1998, 2001).

Note that an open source product in the categories discussed here need not be a full software product. It can be a more limited group of modules or classes for existing open source software. For example, IBM initiates and supports various Apache-related projects and releases related code even when it is developed in-house. IBM's main source of revenue in this case is derived from selling Web servers, which benefit greatly from improvements and additions to Apache.

We characterize price, quality, and hiring paths for firms under both the open source and closed source models. One interesting finding is that under both open and closed source, the software will in many instances initially be free (also known as freeware) if there is no lower bound on the price. Otherwise it will be at its lower bound. In this paper, we assume the lower bound to be zero. That is, if the firm wishes to provide a software product free of charge to boost sales of another product, it may not need to open the source. Interestingly, if the firm elects to close its source, the higher the initial quality of the complementary good, and the longer the software will be free.

The optimal decision on opening the source also depends on the importance of user contributions, the wages and effectiveness of in-house developers, and the initial qualities of the products. When a firm decides to pursue a software project, it will generally have an in-house developed prototype and some assessment of the in-house development potential.

If the potential contribution from outside programmers is not perceived to be large, obviously the firm should not pursue open source. Furthermore, when the benefit to the firm's complementary commercial product is small relative to the value of the in-house developed software to the consumers, the firm should keep the software product proprietary and extract as much of the consumer surplus as it can. Finally, welfare implications to society should be considered. Although it may be optimal for the firm to close the source code, for finitely lived software, open source improves society's welfare in terms of both quality and productivity.

The solution approach taken here is that of optimal control over a finite horizon. This approach is ideal because the firm is able to vary the price of its commercial products over time and because related variables (state variables) such as quality and network size also vary over time. We believe a finite horizon is appropriate as technology products typically have finite lives (e.g., Norton and Bass 1987) and firms plan in advance for the birth of new generations and the death of old ones. We assume software lifetimes can be roughly predicted (e.g., Tamai and Torimitsu 1992).

Such an approach has been often taken in the literature to address dynamic pricing problems (e.g., Chintagunta and Rao 1996; Gaimon 1986, 1988, 1989; Gallego and van Ryzin 1994; Elmaghraby and Keskinocak 2003) and dynamic quality problems (Carrillo and Gaimon 2000, Fine 1986, Fine and Li 1988, Mukhopadhyay and Kouvelis 1997, Kouvelis et al. 1997, Kouvelis and Mukhopadhyay 1999, Muller and Peles 1988). The approach presented here allows for dynamic pricing and quality in the presence of network externalities. A network effect, or network externality (Katz and Shapiro 1985, 1986), is the idea that utility from a product is increasing in the number of other users. It is a pervasive feature of the markets for software and other information goods (Shapiro and Varian 1999). A network effect implies that the larger the network of existing users, the more likely nonadopters are to adopt.

Also note the parallel between the present paper and some works in the innovation diffusion literature. Diffusion is generally defined as the process by which information about an innovation is communicated through a social system (Rogers 1983). In essence, diffusion can be thought of a special type of network effect that, when accounting for the dependence between demand and price, translates to a dynamic pricing problem for a product (e.g., Kalish 1983, Kalish and Lilien 1983, Nascimento and Vanhonacker 1988, Sethi and Bass 2003).

The approach we take here expands on the above dynamic pricing literature by examining a scenario of two complementary goods, one of which can be developed through user contributions contingent on it being free. The paper is organized as follows: §2 lays out the basic model and assumptions. Section 3 develops the theory and its analytical implications as they relate to dynamics in finite horizon setting. Section 4 pursues numerical simulations for comparative statics and further insights. Section 5 concludes. The appendix (available as an Online Supplement at http://www.poms.org/journal/supplements) contains all the proofs not included in the text.

2. The Models and Setting

The firm has two products, product 1 and product 2, which have different codependent demand rates, D_1 and D_2, respectively. In the case of closed source development, all quality improvements arise in-house as a function of the number of in-house developers (e.g., Cohen et al. 1996, Joglekar et al. 2001). In the case of open source development, all quality improvements come from the users, as a function of the size m of the network.

$Q_1(t) =$ The quality of the software at time t.
$Q_2(t) =$ The exogenously given quality of the complementary product at time t.
$m(t) =$ Size of network of users (i.e., installed base) at time t.
$N(t) =$ Number of in-house developers at time t.
$P_1(t) =$ The price of the software at time t.
$P_2(t) =$ The price of the complementary product at time t.

The profit in a given period is the revenue P_1D_1 from the software, plus the revenue P_2D_2 from the complementary product, minus development costs if applicable and minus adaptation cost of the software to the firm's other products. This adaptation cost is particularly relevant in the open source case, where the source code is not necessarily developed with the firm's objectives in mind. In the open source case, P_1 is by definition zero, and so the revenue is only P_2D_2. This loss of revenue may be offset partly by the fact that development costs are zero. This is not to say that the firm incurs no additional cost in adapting the open source software to its products. Indeed, IBM is reported to have spent $1 billion on open source projects in 2001 alone (Lerner and Tirole 2005). We capture this additional cost by $C_O \geq 0$ in the model. C_O is assumed to be a fixed cost. Note that C_O is the additional cost involved in adaptation of the open source code to the firm's other products. We assume that in the case of closed source, the code is written to be compatible with the firm's other products, and so the cost of ensuring this compatibility is negligible. In some cases, there might be a cost to ensuring compatibility even in the closed source case. If so, C_O could be interpreted as the difference between the cost of

adaptation in open source case and the parallel cost in the closed source case.

In the closed source case, development costs increase with the number of in-house developers, N. We assume a cost function of wN^2 in accordance with the economic principle of increasing marginal cost, where w is not unit wage but rather a cost parameter. One can think of a generalized cost function wN^γ. Given the law of increasing marginal costs, $\gamma = 2$ is a reasonable example. We also assume there is no hiring or firing cost associated with labor decisions. Note that in workforce planning models (e.g., Gaimon 1997), employment policies such as hiring and firing are critical, but such issues are outside the scope of the present work. In the closed source case, the in-house contributors improve the quality of the software over time. That is,

$$\dot{Q}_1 = kN - \delta Q_1, \quad Q_1(0) = Q_1^0 \geq 0. \tag{1}$$

The parameter $k > 0$ denotes the productivity or effectiveness of the in-house closed source programmers. When k is high, closed source programmers are very effective and open source development may not be warranted.

Note that software quality depreciates in our model. Clearly, measures of quality such as lines of code or other code-based measures do not decline. However, here software quality is a term in the demand function. From a demand perspective, if the environment is moving forward and the software is not, then its relative quality (represented as "quality" in the demand function) is declining. In Martin Fink's (2003, p. 166) book *The Business and Economics of Linux and Open Source*, he makes the point that given time, the value of software eventually decreases through a process of "devaluation [which] comes from competitive forces and a somewhat natural commodity effect." Tamai and Torimitsu (1992, p. 63) state that it is common knowledge that "software keeps on functionally evolving but structurally deteriorating." We assume that the quality becomes obsolete over time at a constant proportional rate $0 < \delta < 1$. This means high quality becomes obsolete faster than low quality. If high quality is defined as being in the forefront of technology, then it is reasonable to assume that maintaining this technological edge is more difficult than maintaining a lower technological standard. For a similar discussion regarding the relationship between depreciation and the rate of obsolescence, with a functional form for the change in the state variable) that is identical to ours, see Southwick and Zionts (1974). In their paper, the state variable is education level, which like software quality does not deteriorate but does become outdated. It is assumed that the initial quality of the software is nonnegative. Because the source is closed to the public, contributions by users are not possible. In the open source case, on the other hand, there are no in-house developers and all contribution is through users. Hence,

$$\dot{Q}_1 = \alpha m - \delta Q_1, \quad Q_1(0) = Q_1^0 \geq 0. \tag{2}$$

The parameter $\alpha > 0$ represents the level of involvement by the open source user community, which includes users of both the software and of the complementary product.

In addition to software quality $Q_1(t)$, we also consider the quality of the complementary product $Q_2(t)$. We allow $Q_2(t)$ to be a dynamic variable as long as it is nonnegative. We assume, however, that $Q_2(t)$ is exogenous to this problem, although it may be endogenous to the firm through variables not considered here.

The size m of the network of users increases each period by the number of new users of both products and decreases by a percentage of the existing users discontinuing the use of the product. The users of both products need not be equally weighted. A user of the software may be more or less valuable to the network than a user of the complementary product. The parameter $a > 0$ measures this relative weight. We also assume separability between demands for the two products in the network, though the two are complements. That is, a user who uses both products has the weight of $(1 + a)$ in the network. The parameter $0 < \varepsilon < 1$ is the rate of depreciation of the network or the rate of exit. Note that the network growth rate is derived from a summation of two implicit separate networks, m_1 and m_2, where m_1 depends on D_1 and m_2 depends on D_2. Each network could conceivably have a different exit rate, denoted ε_1 and ε_2, respectively. However, for ease of exposition we assume the exit rates to be the same, resulting in a single ε. Hence,

$$\dot{m} = aD_1 + D_2 - \varepsilon m, \quad m(0) = m^0 \geq 0. \tag{3}$$

The demand of the complementary product is $D_2 = h(P_2, m, Q_2)$, where $h(P_2, m, Q_2) \geq 0$, $\partial h/\partial P_2 \leq 0$, $\partial h/\partial m \geq 0$, and $\partial h/\partial Q_2 \geq 0$. Demand D_2 is decreasing in the price of the complementary product P_2. This is due to the law of demand, which states that price and demand are inversely related. Demand D_2 is increasing in the size m of the network of users. D_2 is affected by the network because of positive network externalities. That is, consumers derive utilities from other consumers using the product through newsgroups, file sharing, increased service, and compatible goods. Finally, demand D_2 is increasing in the quality Q_2 of the complementary product. This is based on the assumption that people derive utility from quality.

We assume that the demand of the software is $D_1 = D_2 g(P_1, Q_1) = h(P_2, m, Q_2) g(P_1, Q_1)$, where

Table 1 The Models

Open source	Closed source
Max $\left\{\int_0^T e^{-\rho t}(P_2 D_2)\,dt - C_O + \sigma(Q_1(T), m(T))e^{-\rho T}\right\}$	Max $\left\{\int_0^T e^{-\rho t}(P_1 D_1 + P_2 D_2 - wN^2)\,dt + \sigma(Q_1(T), m(T))e^{-\rho T}\right\}$
s.t. $\dot{Q}_1 = \alpha m - \delta Q_1, \quad Q_1(0) = Q_1^0 \geq 0$ $\dot{m} = aD_1 + D_2 - \varepsilon m, \quad m(0) = m^0 \geq 0$ $P_2 \geq 0$	s.t. $\dot{Q}_1 = kN - \delta Q_1, \quad Q_1(0) = Q_1^0 \geq 0$ $\dot{m} = aD_1 + D_2 - \varepsilon m, \quad m(0) = m^0 \geq 0$ $P_1, P_2, N \geq 0$
where $D_1 = D_2 g(0, Q_1) \geq 0, \quad D_2 = h(P_2, m, Q_2) \geq 0, \quad \frac{\partial \sigma}{\partial Q_1}, \frac{\partial \sigma}{\partial m} \geq 0$	where $D_1 = D_2 g(P_1, Q_1) \geq 0, \quad D_2 = h(P_2, m, Q_2) \geq 0, \quad \frac{\partial \sigma}{\partial Q_1}, \frac{\partial \sigma}{\partial m} \geq 0$

$g(P_1, Q_1) \geq 0$, $\partial g/\partial P_1 \leq 0$, and $\partial g/\partial Q_1 \geq 0$. This means that D_1 decreases as the price of the software P_1 increases and increases as the quality of the software Q_1 increases. The motivation for these first-order relationships is similar to that of D_2, as described above. In the open source case, as $P_1 = 0$, then $D_1 = D_2 g(0, Q_1)$. The multiplication of the function $g(P_1, Q_1)$ by D_2 represents the effect of complementarity. That is, the demand for the software productincreases with the demand for the complementary product. At this point, some more discussion is warranted on the relationship between D_1 (demand for software) and D_2 (demand for the complement). D_1 and D_2 affect each other, but in different ways. D_1 is affected by D_2 directly through complementarities. That is, a portion of the complement's users (D_2) may wish to use the software (D_1). However, the software users can use the software independently of the complementary good (they could substitute any number of other products). D_2 is affected by D_1 indirectly through the network size (which is a function of purchases of both 1 and 2) and by the network directly through network externalities (as explained above). D_1 is not affected by the network directly. However, in the open source case, D_1 is affected by the network indirectly through the impact of the network on the quality of the code, Q_1.

Suppose the products, both the software and the complementary product, have finite lives with a known terminal period T ($0 < T < \infty$). The terminal period could be due to an anticipated release of a new generation of products or technologies or due to a known date for the firm to cease operations. We assume the salvage value of both products is $\sigma(Q_1(T), m(T))$, which is a function of the ending state variables at time T. Because we assume no hiring and layoff costs, it is safe to assume that N has zero salvage value at the end of the product's life. Zero salvage value means that the terminal value $N(T)$ can be left equal to $N(T^-)$. If there is a positive salvage value for $N(T)$, then we could hire infinitely many people at T because there is no hiring cost and because there are no wages to be paid beyond T. If there is negative salvage value for $N(T)$ in terms of future wages to be paid, then we would lay off $N(T^-)$ workers all at once at time T to have $N(T) = 0$. We also assume that $\partial\sigma/\partial Q_1 \geq 0$ and $\partial\sigma/\partial m \geq 0$.

We summarize the two models in Table 1.

Note that the initial values of the state variables are nonnegative, $Q_1^0 \geq 0$ and $m^0 \geq 0$. This will be important for the proofs of the nonnegative trajectories of $Q_1(t)$ and $m(t)$. It is also realistic because product quality and the size of user network cannot in reality be negative. We also restrict the prices and the number of developers to have lower bounds at zero, $P_1(t) \geq 0$, $P_2(t) \geq 0$ and $N(t) \geq 0$. This is a fair constraint if subsidies are not allowed. In our simulations, to be described shortly, the price constraints are binding at the beginning of the product life for both open and closed source cases. The nonnegativity constraint on the number of developers is only binding at time T if the salvage value $\sigma(Q_1(T), m(T))$ is zero.

3. Analytical Investigation

3.1. Open Source

The firm maximizes its discounted (discount rate is $\rho > 0$) profit stream over the time interval $0 \leq t \leq T$, plus the discounted salvage value of both products at time T, and minus the adaptation cost of the software to the firm's other products. The instantaneous profit rate consists of the price for the commercial product (the price for the open source software product, P_1, is zero) multiplied by the demand for the commercial product at each period. In this problem, $P_2(t)$ is the control variable at time t. We let $J(P_2(\cdot))$ denote the objective function that the firm maximizes with respect to the control variable P_2. We let $V_O(0, Q_1(0), m(0))$ denote the value function starting at time 0 in state $Q_1(0)$ and $m(0)$. Note that $V_O(0, Q_1(0), m(0))$ is merely the maximized value of $J(P_2(\cdot))$. It will be useful to compare the open source to the closed source later on.

$$V_O(0, Q_1(0), m(0)) = \max\left\{J(P_2(\cdot)) = -C_O + \int_0^T P_2 h(P_2, m, Q_2) e^{-\rho t}\,dt + \sigma(Q_1(T), m(T))e^{-\rho T}\right\} \quad (4)$$

$$\text{s.t.} \quad \dot{Q}_1 = \alpha m - \delta Q_1, \quad Q_1(0) = Q_1^0, \tag{5}$$

$$\dot{m} = ah(P_2, m, Q_2)g(0, Q_1) + h(P_2, m, Q_2) - \varepsilon m, \quad m(0) = m^0, \tag{6}$$

$$P_2 \geq 0. \tag{7}$$

We assume that $F_O \equiv P_2 h(P_2, m, Q_2)$ is concave in P_2. This assumption holds for two common demand functions we use later in the paper. We form the Lagrangian:

$$L = P_2 h(P_2, m, Q_2) + \lambda(\alpha m - \delta Q_1) + \mu\big[ah(P_2, m, Q_2)g(0, Q_1) + h(P_2, m, Q_2) - \varepsilon m\big] + \eta_2 P_2. \tag{8}$$

From this we get the adjoint equations

$$\dot{\lambda} = \rho\lambda - \frac{\partial L}{\partial Q_1} = (\rho + \delta)\lambda - a\mu h(P_2, m, Q_2)\frac{\partial g}{\partial Q_1}, \quad \lambda(T) = \frac{\partial \sigma}{\partial Q_1}\bigg|_T, \tag{9}$$

$$\dot{\mu} = \rho\mu - \frac{\partial L}{\partial m} = (\rho + \varepsilon)\mu - \alpha\lambda - [P_2 + \mu(ag(0, Q_1) + 1)]\frac{\partial h}{\partial m}, \quad \mu(T) = \frac{\partial \sigma}{\partial m}\bigg|_T. \tag{10}$$

The optimal control must satisfy

$$\frac{\partial L}{\partial P_2} = h(P_2, m, Q_2) + [P_2 + \mu(ag(0, Q_1) + 1)]\frac{\partial h}{\partial P_2} + \eta_2 = 0, \tag{11}$$

and the Lagrange multiplier η_2 must satisfy the complementary slackness condition

$$\eta_2 \geq 0, \quad \eta_2 P_2 = 0. \tag{12}$$

Proposition 3.1. *In the open source case, $Q_1(t) \geq 0$ and $m(t) \geq 0$ for $0 \leq t \leq T$. Moreover, these variables are strictly positive if their initial values $Q_1^0 > 0$ and $m^0 > 0$.*

Proposition 3.2. *In the open source case, the optimal profit* (a) *increases with α*; (b) *increases with the initial software quality Q_1^0.*

The variable λ is interpreted as the per unit change in the value function for small changes in the software quality Q_1. In other words, $\lambda(t)$ is the marginal value per unit of software quality at time t. Similarly, the variable μ is interpreted as the per unit change in the value function for small changes in the size of the network m. That is, $\mu(t)$ is the marginal value per unit of the network at time t. The next two propositions show that λ and μ are nonnegative. These propositions are important because they show that the firm would benefit from higher software quality and network size.

Proposition 3.3. *In the open source model, the marginal benefit λ of increasing the software quality is nonnegative.*

Proposition 3.4. *In the open source model, the marginal benefit μ of increasing the size of the network is nonnegative.*

We next turn to examining prices. We make a distinction between myopic prices, $\hat{P}_2(t)$, and forward-looking prices, $P_2^*(t)$. Specifically, myopic prices maximize immediate returns without future consideration, whereas forward-looking prices maximize returns over the entire horizon. We show that myopic prices are always excessive relative to forward-looking prices. Recall that $F_O(t) = P_2(t)h(P_2(t), m(t), Q_2(t))$ is the revenue accrued at time t. Let $\hat{P}_2(m(t), Q_2(t)) = \arg\max_{P_2(t)} F_O(t)$. $P_2^*(t)$ is the solution to (4). Let $m^*(t)$ be the optimal network size corresponding to $P_2^*(t)$ at time t.

Proposition 3.5. *$P_2^*(t) \leq \hat{P}_2(m^*(t), Q_2(t))$ for $0 \leq t \leq T$. Moreover, $P_2^*(T) = \hat{P}_2(m^*(T), Q_2(T))$ if the salvage value is zero at time T.*

Proposition 3.5 is important for several reasons. First, it implies that myopic behavior results in excessive prices. Second, it allows us later on to arrive at an upper bound for the commercial product's price.

3.2. Closed Source

The firm maximizes its discounted profit stream over the time interval $0 \leq t \leq T$, plus the discounted salvage value of both products at time T. The instantaneous profit rate is the revenue of the software and the commercial product minus the development costs at each period. In this problem, $P_1(t)$, $P_2(t)$, and $N(t)$ are the control variables at time t. $J(P_1(\cdot), P_2(\cdot), N(\cdot))$ is the objective function. $V_C(0, Q_1(0), m(0))$ is the value function given that we start at time 0 in state $Q_1(0)$ and $m(0)$.

$$V_C(0, Q_1(0), m(0)) = \max\Big\{J(P_1(\cdot), P_2(\cdot), N(\cdot)) = \int_0^T \big[P_1 h(P_2, m, Q_2)g(P_1, Q_1) + P_2 h(P_2, m, Q_2) - wN^2\big]e^{-\rho t}\,dt + \sigma(Q_1(T), m(T))e^{-\rho T}\Big\} \tag{13}$$

$$\text{s.t.} \quad \dot{Q}_1 = kN - \delta Q_1, \quad Q_1(0) = Q_1^0, \tag{14}$$

$$\dot{m} = ah(P_2, m, Q_2)g(0, Q_1) + h(P_2, m, Q_2) - \varepsilon m, \quad m(0) = m^0, \tag{15}$$

$$P_1, P_2 \geq 0, \quad N \geq 0. \tag{16}$$

We assume that $F_C \equiv P_1 h(P_2, m, Q_2) g(P_1, Q_1) + P_2 h(P_2, m, Q_2) - wN^2$ is jointly concave in P_1, P_2, and N. We form the Lagrangian:

$$\begin{aligned} L = {} & P_1 h(P_2, m, Q_2) g(P_1, Q_1) + P_2 h(P_2, m, Q_2) \\ & - wN^2 + \lambda(kN - \delta Q_1) \\ & + \mu[ah(P_2, m, Q_2) g(P_1, Q_1) + h(P_2, m, Q_2) - \varepsilon m] \\ & + \eta_1 P_1 + \eta_2 P_2 + \eta_3 N. \end{aligned} \tag{17}$$

From this we get the adjoint equations

$$\begin{aligned} \dot{\lambda} &= \rho\lambda - \frac{\partial L}{\partial Q_1} \\ &= (\rho + \delta)\lambda - (P_1 + a\mu) h(P_2, m, Q_2) \frac{\partial g}{\partial Q_1}, \\ \lambda(T) &= \left. \frac{\partial \sigma}{\partial Q_1} \right|_T, \end{aligned} \tag{18}$$

$$\begin{aligned} \dot{\mu} &= \rho\mu - \frac{\partial L}{\partial m} \\ &= (\rho + \varepsilon)\mu - \left[P_2 + \mu + (P_1 + a\mu) g(P_1, Q_1)\right] \frac{\partial h}{\partial m}, \\ \mu(T) &= \left. \frac{\partial \sigma}{\partial m} \right|_T. \end{aligned} \tag{19}$$

The optimal control must satisfy

$$\begin{aligned} \frac{\partial L}{\partial P_1} = {} & \left[g(P_1, Q_1) + (P_1 + a\mu) \frac{\partial g}{\partial P_1} \right] \\ & \cdot h(P_2, m, Q_2) + \eta_1 = 0, \end{aligned} \tag{20}$$

$$\begin{aligned} \frac{\partial L}{\partial P_2} = {} & h(P_2, m, Q_2) + [P_2 + \mu + (P_1 + a\mu) g(P_1, Q_1)] \\ & \times \frac{\partial h}{\partial P_2} + \eta_2 = 0, \end{aligned} \tag{21}$$

$$\frac{\partial L}{\partial N} = -2wN + k\lambda + \eta_3 = 0, \tag{22}$$

and the Lagrange multiplier η_1, η_2, and η_3 must satisfy the complementary slackness condition

$$\eta_1 \geq 0, \quad \eta_1 P_1 = 0, \tag{23}$$

$$\eta_2 \geq 0, \quad \eta_2 P_2 = 0, \tag{24}$$

$$\eta_3 \geq 0, \quad \eta_3 N = 0. \tag{25}$$

Proposition 3.6. *In the closed source case, $Q_1(t) \geq 0$ and $m(t) \geq 0$ for $0 \leq t \leq T$. Moreover, these variables are strictly positive if their initial values $Q_1^0 > 0$ and $m^0 > 0$.*

Proposition 3.7. *In the closed source case, the optimal profit* (a) *increases with k*; (b) *increases with the initial software quality Q_1^0*; (c) *decreases with w.*

The next two propositions show that the firm benefits from higher software quality and higher network size.

Proposition 3.8. *In the closed source model, the marginal benefit λ of increasing the software quality is nonnegative.*

Proposition 3.9. *In the closed source model, the marginal benefit μ of increasing the size of the network is nonnegative.*

As in the open source analysis, we make a distinction between myopic prices, $\hat{P}_1(t)$ and $\hat{P}_2(t)$, and forward-looking prices, $P_1^*(t)$ and $P_2^*(t)$, where myopic prices maximize immediate returns without future consideration and forward-looking prices maximize returns over the entire horizon. We again show that myopic prices are always excessive relative to forward-looking prices. Recall that $F_C(t) = P_1(t)h(P_2(t), m(t), Q_2(t))g(P_1(t), Q_1(t)) + P_2(t)h(P_2(t), m(t), Q_2(t)) - wN(t)^2$ is the profit accrued at time t. Let $(\hat{P}_1(m(t), Q_1(t), Q_2(t)), \hat{P}_2(m(t), Q_1(t), Q_2(t))) = \arg\max_{P_1(t), P_2(t)} F_C(t)$. $(P_1^*(t), P_2^*(t))$ is the solution to (13). Let $m^*(t)$ and $Q_1^*(t)$ be the optimal network size and software quality corresponding to $(P_1^*(t), P_2^*(t))$ at time t.

Proposition 3.10. *$P_1^*(t) \leq \hat{P}_1(m^*(t), Q_1^*(t), Q_2(t))$ and $P_2^*(t) \leq \hat{P}_2(m^*(t), Q_1^*(t), Q_2(t))$ for $0 \leq t \leq T$. Moreover $P_1^*(T) = \hat{P}_1(m^*(T), Q_1^*(T), Q_2(T))$ and $P_2^*(T) = \hat{P}_2(m^*(T), Q_1^*(T), Q_2(T))$ if the salvage value is zero at time T.*

4. Numerical Analysis

We examine different scenarios numerically in both open source and closed source models. We assume $\sigma(Q_1(T), m(T)) = \upsilon Q_1(T) + \varphi m(T)$, where υ and φ are nonnegative constants. Therefore, $\lambda(T) = \upsilon$ and $\mu(T) = \varphi$. To simplify the calculation, we assume Q_2 is positive and constant for $0 \leq t \leq T$.

We examine two types of demand functions. The first one is the negative exponential demand function, which has several desirable properties (see Greenhut and Greenhut 1977, Anderson 1989, Haruvy et al. 2003). One nice property is that it precludes the possibility of negative demand. The demand function of the complementary product is

$$D_2 = h(P_2, m, Q_2) = \exp\left(-\frac{P_2}{mQ_2}\right), \tag{26}$$

and the demand of the software is

$$D_1 = D_2 g(P_1, Q_1) = \exp\left(-\frac{P_2}{mQ_2}\right) \exp\left(-\frac{P_1 + c}{Q_1}\right). \tag{27}$$

Because the price P_1 for the open source software product is zero, in the open source case the demand of the software is

$$D_1 = D_2 g(0, Q_1) = \exp\left(-\frac{P_2}{mQ_2}\right) \exp\left(-\frac{c}{Q_1}\right). \tag{28}$$

The positive constant c in Equations (27) and (28) is necessary to keep the numerator strictly above zero in the open source case. Without the positive constant c, demand D_1 would equal D_2 in the open source case.

The second demand function we examine is linear in price. The demand function of the complementary product is

$$D_2 = h(P_2, m, Q_2) = B - \frac{P_2}{mQ_2}, \quad B - \frac{P_2}{mQ_2} \geq 0, \quad (29)$$

and the demand of the software is

$$D_1 = D_2 g(P_1, Q_1) = \left(B - \frac{P_2}{mQ_2}\right)\left(A - \frac{P_1 + c}{Q_1}\right), \quad \left(A - \frac{P_1 + c}{Q_1}\right) \geq 0. \quad (30)$$

In the open source case the demand of the software is

$$D_1 = D_2 g(0, Q_1) = \left(B - \frac{P_2}{mQ_2}\right)\left(A - \frac{c}{Q_1}\right), \quad \left(A - \frac{c}{Q_1}\right) \geq 0. \quad (31)$$

We use Fortran code and Excel spreadsheets to find numerical solutions. Due to computational limitations, numerical solutions are the result of approximations (e.g., Sethi and Thompson 2000, pp. 48–50). Specifically, for each state and adjoint variable x, we substitute $\triangle x/\triangle t = (x(t+\triangle t) - x(t))/\triangle t$ for $\dot{x}$. We let $\triangle t = 0.01$. As a numerical example, we consider the parameters: $\delta = 0.03$, $\varepsilon = 0.01$, $\rho = 0.1$, $a = c = k = w = \alpha = 1$, $Q_1^0 = m^0 = Q_2^0 = 1$, $\upsilon = \varphi = 0$, $C_O = 0$, and $T = 50$. These parameter values will be used throughout this section in all the illustrations that follow. Note also that by Propositions 3.1 and 3.6, $Q_1(t) > 0$ and $m(t) > 0$, for $0 \leq t \leq T$.

By Propositions 3.5 and 3.10, we know that prices have upper bounds defined by the myopic prices. The next two propositions provide the upper bounds for both exponential demand function and linear-price demand function. Figure 1 shows the prices and their bounds.

We define "soft" and "hard" upper bounds. A soft upper bound depends on the trajectories of other variables. A hard upper bound is independent of the trajectories of all other state and control variables. A soft upper bound can serve as a reference point so that we can indicate a trajectory's position relative to trajectories of other variables. A hard upper bound is useful as an ex ante indication of a range of values a variable can take.

To make our presentation easy, we define $\hat{P}_1^*(t) \equiv \hat{P}_1(m^*(t), Q_1^*(t), Q_2(t))$ and $\hat{P}_2^*(t) \equiv \hat{P}_2(m^*(t), Q_1^*(t), Q_2(t))$.

Proposition 4.1. (i) *In the open source model with exponential demand function, the optimal price of the complementary product $P_2^*(t)$ has a soft upper bound of $\hat{P}_2^*(t) = m^*(t)Q_2(t)$.*

(ii) *For the linear-price demand function, the optimal price of the complementary product $P_2^*(t)$ has a soft upper bound of $\hat{P}_2^*(t) = \frac{1}{2}Bm^*(t)Q_2(t)$.*

Proof. This result follows immediately from Proposition 3.5 and the demand functions of $h(P_2, m, Q_2)$. □

Note that the soft upper bound is the myopic optimal path. We can also define a hard upper bound for P_2. Because m has a hard upper bound, so does P_2.

Corollary 4.1. (i) *In the open source model with exponential demand function, the optimal price of the complementary product $P_2^*(t)$ has a hard upper bound $\overline{m}(t)Q_2(t)$, where $\overline{m}(t) = m^0e^{-\varepsilon t} + (a+1)(1-e^{-\varepsilon t})/\varepsilon$.*

(ii) *For the linear-price demand function, the optimal price of the complementary product $P_2^*(t)$ has a hard upper bound $\frac{1}{2}B\overline{m}(t)Q_2(t)$, where $\overline{m}(t) = m^0e^{-\varepsilon t} + (aA+1) \cdot B(1-e^{-\varepsilon t})/\varepsilon$.*

Proposition 4.2. (i) *In the closed source model with exponential demand function, the optimal price of the software $P_1^*(t) \leq \hat{P}_1^*(t) = Q_1^*(t)$. The optimal price of the complementary product $P_2^*(t) \leq \hat{P}_2^*(t) = m^*(t)Q_2(t) - \hat{P}_1^*(t)g(\hat{P}_1^*(t), Q_1^*(t))$.*

(ii) *For the linear-price demand function, the optimal price of the software $P_1^*(t) \leq \hat{P}_1^*(t) = \frac{1}{2}(AQ_1^*(t) - c)$. The optimal price of the complementary product $P_2^*(t) \leq \hat{P}_2^*(t) = \frac{1}{2}(Bm^*(t)Q_2(t) - \hat{P}_1^*(t)g(\hat{P}_1^*(t), Q_1^*(t)))$.*

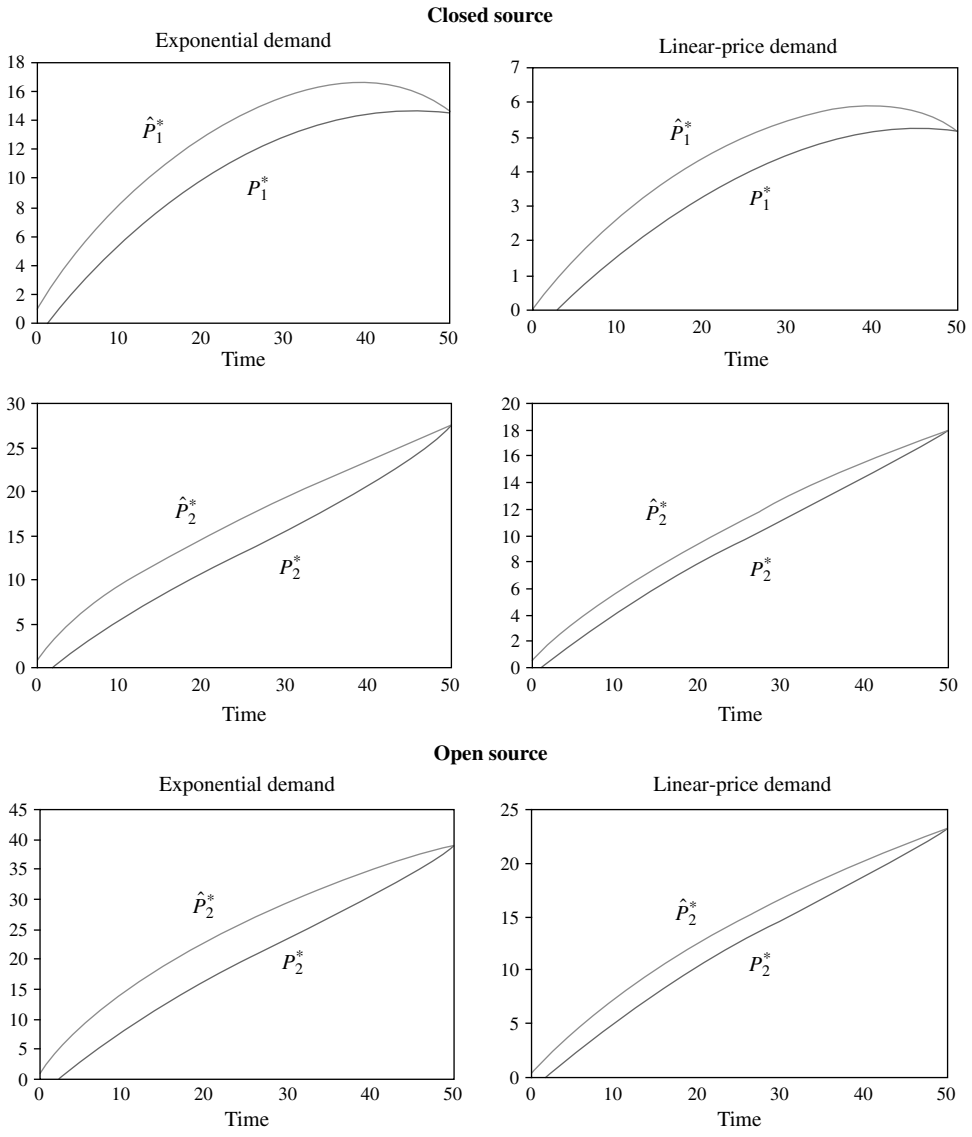
Proof. Result follows immediately from Proposition 3.10 and the demand functions of $h(P_2, m, Q_2)$ and $g(P_1, Q_1)$. □

Corollary 4.2. (i) *In the closed source model with exponential demand function, the optimal price of the software $P_1^*(t)$ has a hard upper bound $\bar{Q}_1(t)$, where $\bar{Q}_1(t) = Q_1^0e^{-\varepsilon t} + kN(0)(1-e^{-\varepsilon t})/\delta$ and $N(0)$ is the number of in-house programmers at time 0. The optimal price of the complementary product $P_2^*(t)$ has a hard upper bound $\overline{m}(t)Q_2(t)$, where $\overline{m}(t) = m^0e^{-\varepsilon t} + (a+1)(1-e^{-\varepsilon t})/\varepsilon$.*

(ii) *For the linear-price demand function, the optimal price of the software $P_1^*(t)$ has a hard upper bound $\frac{1}{2}(A\bar{Q}_1(t) - c)$, where $\bar{Q}_1(t) = Q_1^0e^{-\varepsilon t} + kN(0)(1-e^{-\varepsilon t})/\delta$ and $N(0)$ is the number of in-house programmers at time 0. The optimal price of the complementary product $P_2^*(t)$ has a hard upper bound $\frac{1}{2}B\overline{m}(t)Q_2(t)$, where $\overline{m}(t) = m^0e^{-\varepsilon t} + (aA+1)B(1-e^{-\varepsilon t})/\varepsilon$.*

Due to network externalities, prices for both products in the closed source model and the price for the complementary product in the open source model begin low in order to quickly establish the network's installed base and gradually increase as the network size increases, as Figure 1 demonstrates.

Figure 1 Prices and Their Upper Bounds over Time

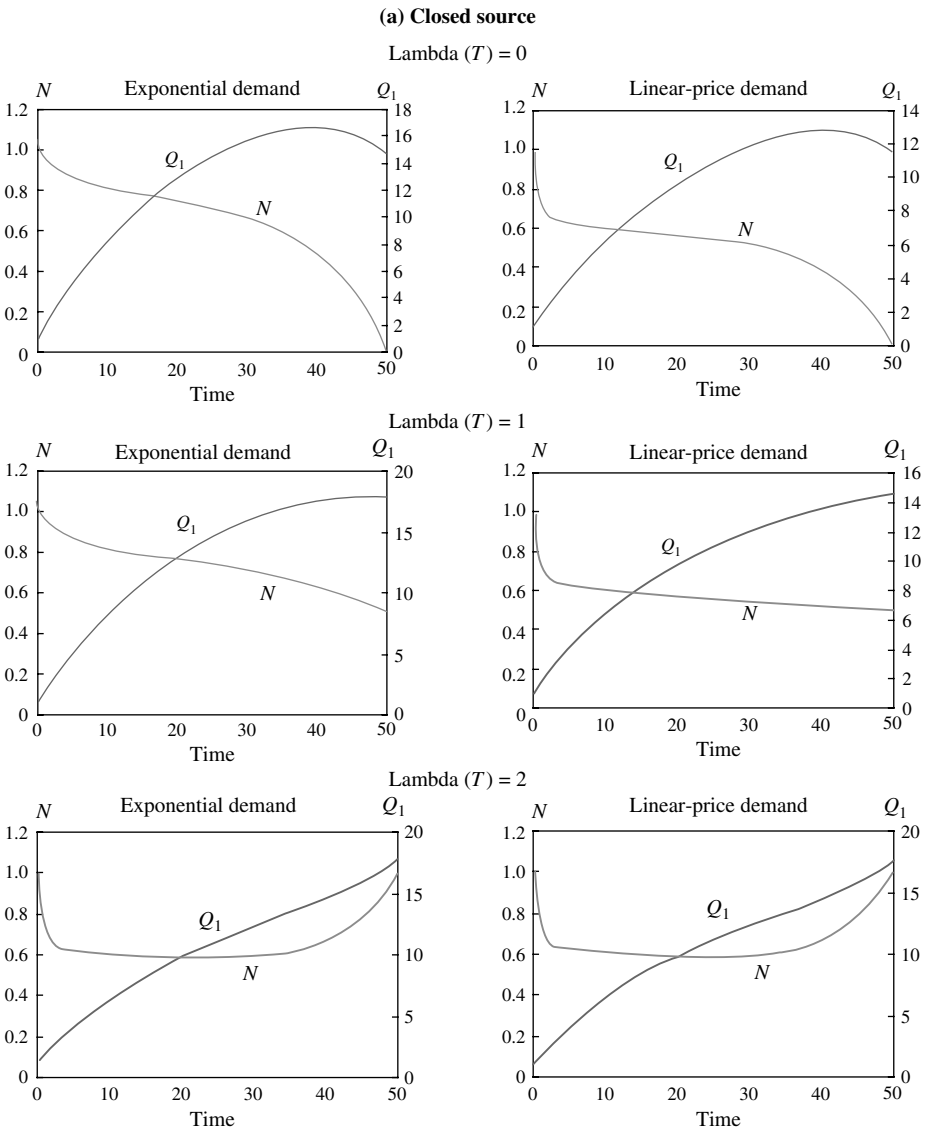


We can see that price in the initial period, for both software and complementary product, is zero. This is because of the network externality effect. To build the network and extract the maximal surplus, the producer is willing to sacrifice short-term profits for larger future profits. Note that the initial zero price is not a general property of all functional forms but rather specific to the (many) cases we study. Nevertheless, we can expect that even in cases where the initial price is positive, it will be low relative to later prices, in order for the network to grow.

The behavior of $P_2(t)$ over time depends on the parameters, and there are several of them. The behavior of $P_2(t)$ is difficult to establish analytically, other than the bounds we show, which are increasing over time. Therefore, we conduct a numerical study of the behavior of $P_2(t)$ over time. For this, we fix $\varepsilon = 0.01$ and $\rho = 0.1$ and examine the behavior of $P_2(t)$ with respect to the remaining parameters.

In the closed source model, hiring is massive early on due to the need to rapidly increase software quality to build up the network. As a result, we see software quality increasing rapidly early in the product's life. With zero salvage value of software quality, the workforce size decreases over time, eventually reaching zero at the end of the product's life. Consequently, software quality begins decreasing towards the end of the life of the product. This is quite intuitive, as given an anticipated product termination and a zero salvage value of software quality at time T, it is not profitable to hire people and improve the quality at the end of the life of the product. When the salvage value of software quality is positive and sufficiently high ($\lambda(T) = 2$ in one of our illustrations in Figure 2), the workforce size will increase towards the end of

Figure 2 Quality and Workforce Size Paths with Different Salvage Values for Software Quality



the product's life. The increased hiring will result in a rapid increase in software quality toward the end of the product's life.

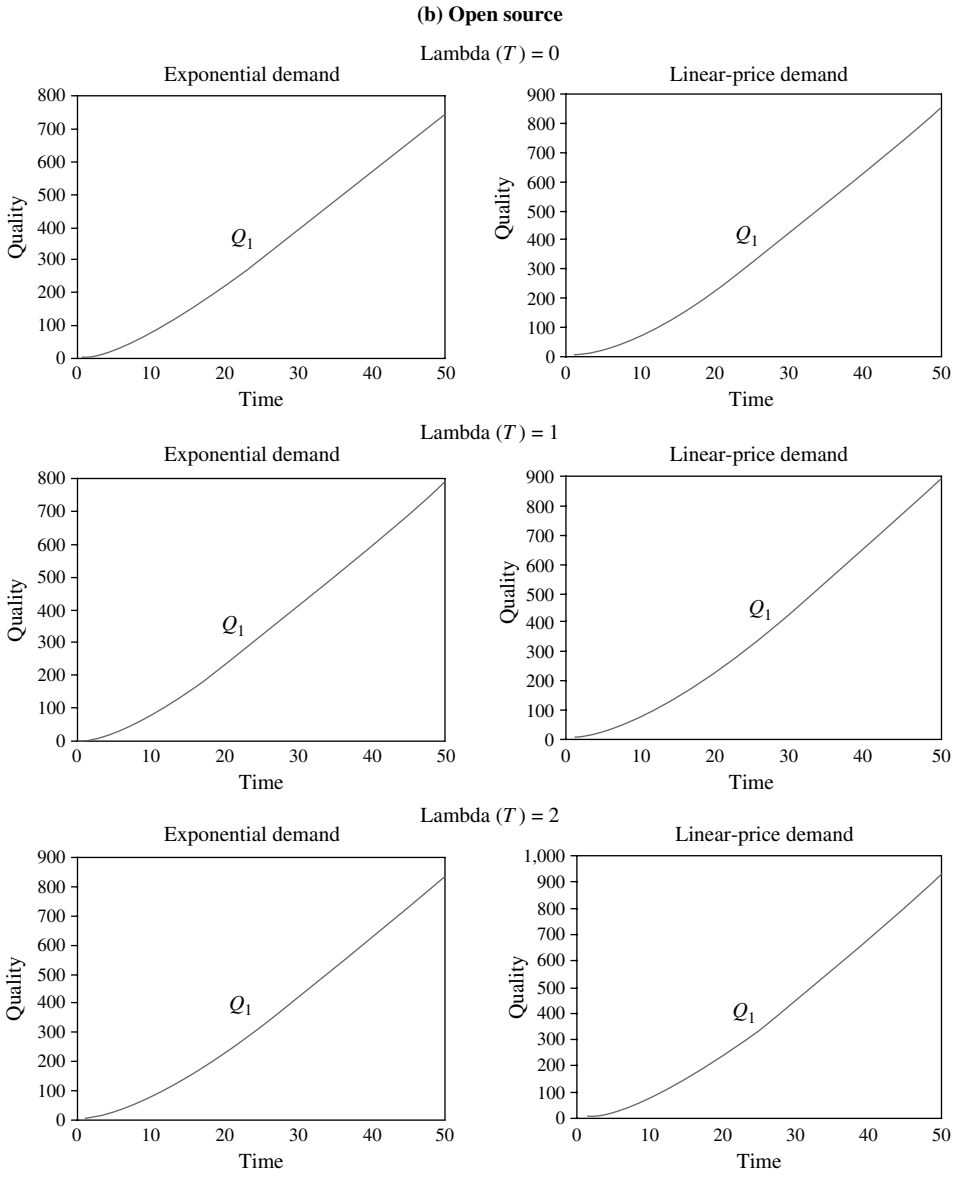
In the open source model, software quality increases continuously because software quality increases as the size of network of users increases. Figure 2 shows quality and workforce size paths with different salvage values for software quality.

An important difference to note between the open source and closed source is that in the closed source case, software quality may decrease toward the end of the life of the software if the salvage value of the software quality is low, whereas in the open source case, it continues to increase for any nonnegative salvage value of software quality. This finding holds for all parameter values. This is because when the software quality's salvage value is low, the firm will not find it optimal to invest in quality when the product nears its death, whereas the open source community, not motivated by profit, will continue to improve the quality. This presents interesting welfare implications to open source development, particularly if the benefit to society from increased quality is substantial.

Next, we illustrate the idea that when a firm has to choose between open source and closed source, that choice may depend on the values of the parameters α, k, w, and the initial quality of the software Q_1^0.

Suppose that the profits for open source and closed source intersect for some value of α. According to Proposition 3.2, part (a), open source profit is monotonically increasing in α and closed source profit is unchanging in α. Therefore, there will be a single point of intersection—a threshold value $\bar{\alpha}$. The numerical analysis clearly confirms that such

Figure 2 Continued



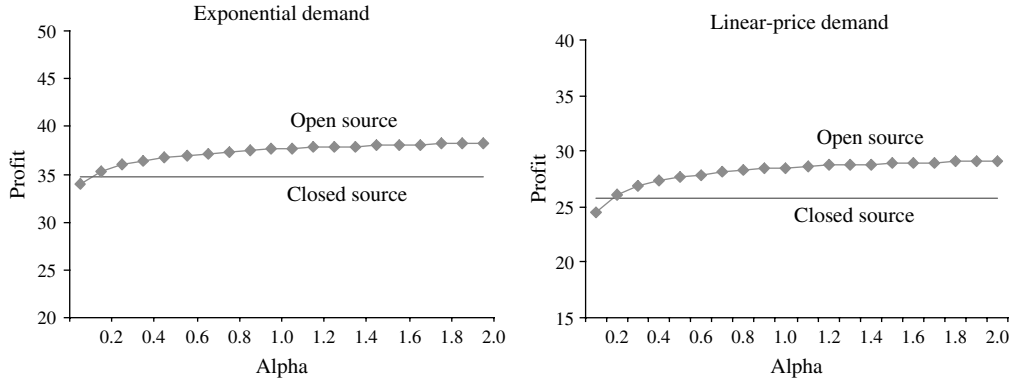
threshold exists: there appears to be a threshold level of open source community involvement, $\bar{\alpha}$, above which open source will be preferred to closed source (see Figure 3). The parameter α is of particular significance to open source developers, as without substantial community development, the software will fail to take off from its initial stage.

In-house programmer productivity is the other side of the productivity coin. Suppose that the profits for open source and closed source intersect for some value of k. According to Proposition 3.7, part (a), closed source profit is monotonically increasing in k and open source profit is unchanging in k. Therefore, there will be a single point of intersection—a threshold in-house programmer productivity, $\bar{k}$, above which closed source will be preferred to open source (see Figure 4).

In considering whether to develop in-house software, the firm must consider wages as well as in-house productivity. Suppose that the profits for open source and closed source intersect for some value of w. According to Proposition 3.7, part (c), closed source profit is monotonically decreasing in w and open source profit is unchanging in w. Therefore, there will be a single point of intersection—a threshold wage factor, $\bar{w}$—above which open source will be preferred to closed source (see Figure 5).

Finally, initial software quality is a critical consideration. According to Proposition 3.2, part (b), and Proposition 3.7, part (b), the optimal profit for both open and closed source models increases with the

Figure 3 Profit over Different Values of α for Open Source and Closed Source



initial software quality Q_1^0. When initial quality of the software is high, the firm has little to benefit from opening the source code to the open source community. As such, there exists a threshold initial quality, Q_1^0, below which open source will be preferred to closed source (see Figure 6). The initial quality of the software appears to have a far greater impact on the closed source profits relative to the open source profits. This is because the software is not sold in the open source case, and so the demand for the software only affects profits indirectly through its effect on the demand for the complementary good.

Moreover, we find that in the closed source case, the higher the initial quality of the complementary product Q_2^0, the longer the price of the software P_1 will remain at zero initially. In addition, the higher the Q_2^0, the higher the P_2 and the lower the P_1 at the end of the time horizon.

From Figure 7, we see that if the quality of the complementary product is high, the firm can charge more for this complementary product and let the software product be free for a longer duration.

5. Conclusions

Whereas the speed and creativity associated with open source development are well accepted, the profit potential is not always recognized. As companies become increasingly involved in open source projects, the profit implications of open source for companies must be better understood. In this article we argue that the profitability of open source software may come from a product that is a complement to the open source code.

A firm considering open source development as an alternative to closed source development would need to review carefully the relationship between the software in question and the firm's other products. If the software is found to enhance the usefulness or quality of complementary products and/or if the users of the software and the users of the complementary product belong to the same network, the complementarity can be exploited. By dynamically and simultaneously managing price, product quality, network size, and hiring for both products, the firm will be able to best exploit the complementarity. This effort may or may not benefit from opening the source of the product. Without a clear model of how to exploit freeware, open source development as a substitute for in-house closed source development may be detrimental to a firm's profitability. Our analytical derivation and simulations demonstrate that under various conditions, open source may not be beneficial to a firm. On the other hand, there are scenarios where the opposite is the case.

Figure 4 Profit over Different Values of k for Open Source and Closed Source

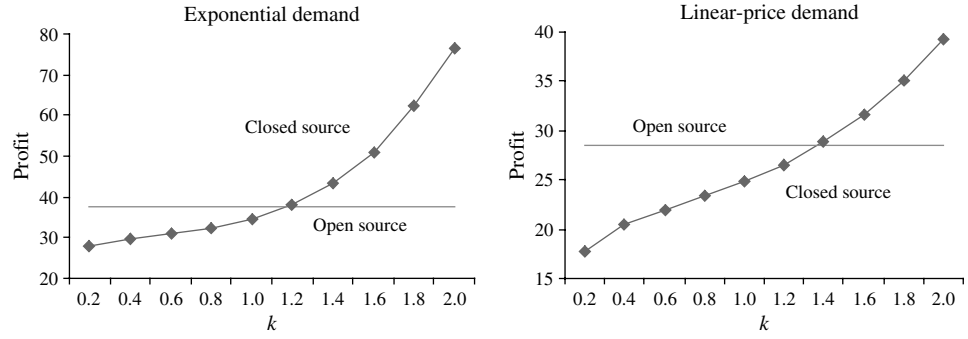
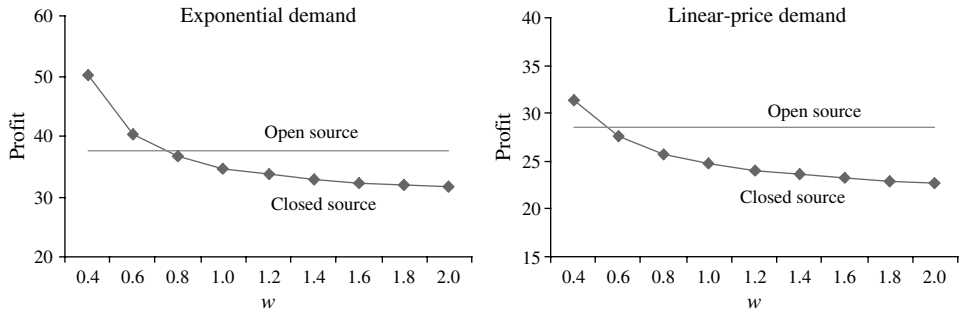


Figure 5 Profit over Different Values of w for Open Source and Closed Source



Specifically, open source community involvement is critical to the success of open source initiative. Only above a critical level of community involvement does open source become a viable alternative to closed source. To the extent that community involvement can be influenced by the firm's efforts or influence, such efforts must complement the pricing decisions evaluated here. In-house programmer productivity was shown to be the opposite side of the same coin. More productive and efficient in-house programmers result in less reliance on open source. However, wage is critical in that respect. If programmer productivity is high but wage is higher than some threshold, our results show that open source would be preferred. Finally, if the initial quality of the software is high, development becomes a less critical consideration and extraction of surplus can begin immediately. In such a case, the firm would prefer to charge a positive price for the software and close the code.

We also characterized price, quality, and hiring paths for firms under both the open source and closed source models. We find that due to network externalities, prices for both products in the closed source model and for the complementary product in the open source model will begin low in order to quickly establish the network's installed base and will gradually increase as the network size increases. We find that in the closed source case, the higher the initial quality of the complementary good, the longer the price of the software will remain at zero initially. That is, when more surplus can be extracted from the complementary good, it may be optimal to keep the software as freeware for a long period of time, even when the source is closed.

In the closed source model, when the salvage value for quality is zero, quality will rapidly increase early on to build the network size and will begin decreasing towards the end of the life of the product. Similarly, hiring will be massive early on and will ease up over time due to the need to increase quality rapidly in order to establish the network. This is not the case in the open source models. As such, it may be argued that for finitely lived products, open source improves society's welfare in terms of both quality and productivity. The problem of software development in the presence of network externalities and complementarities has been examined in other frameworks. However, the optimal control framework is ideally suited to jointly examine optimal trajectories for price, quality and network size as well as for determining the marginal values of quality and network size.

Future research should examine other forms of revenue extraction from open source. From the cost side, future research should investigate more complex expressions for the cost of adoption of open source to the firm's overall product line strategy. The problem can also be studied from a social planner's perspective rather than a firm's perspective. We would further like to examine the effect of competitive pressures on the firm's decision to use open source. When the firm opens its source, it may implicitly provide advantages to competitors (e.g., Zhang et al.

Figure 6 Profit over Levels of Initial Quality Q_1^0

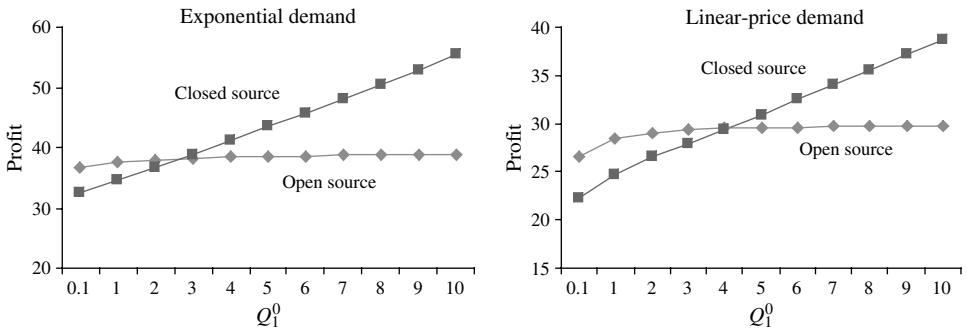
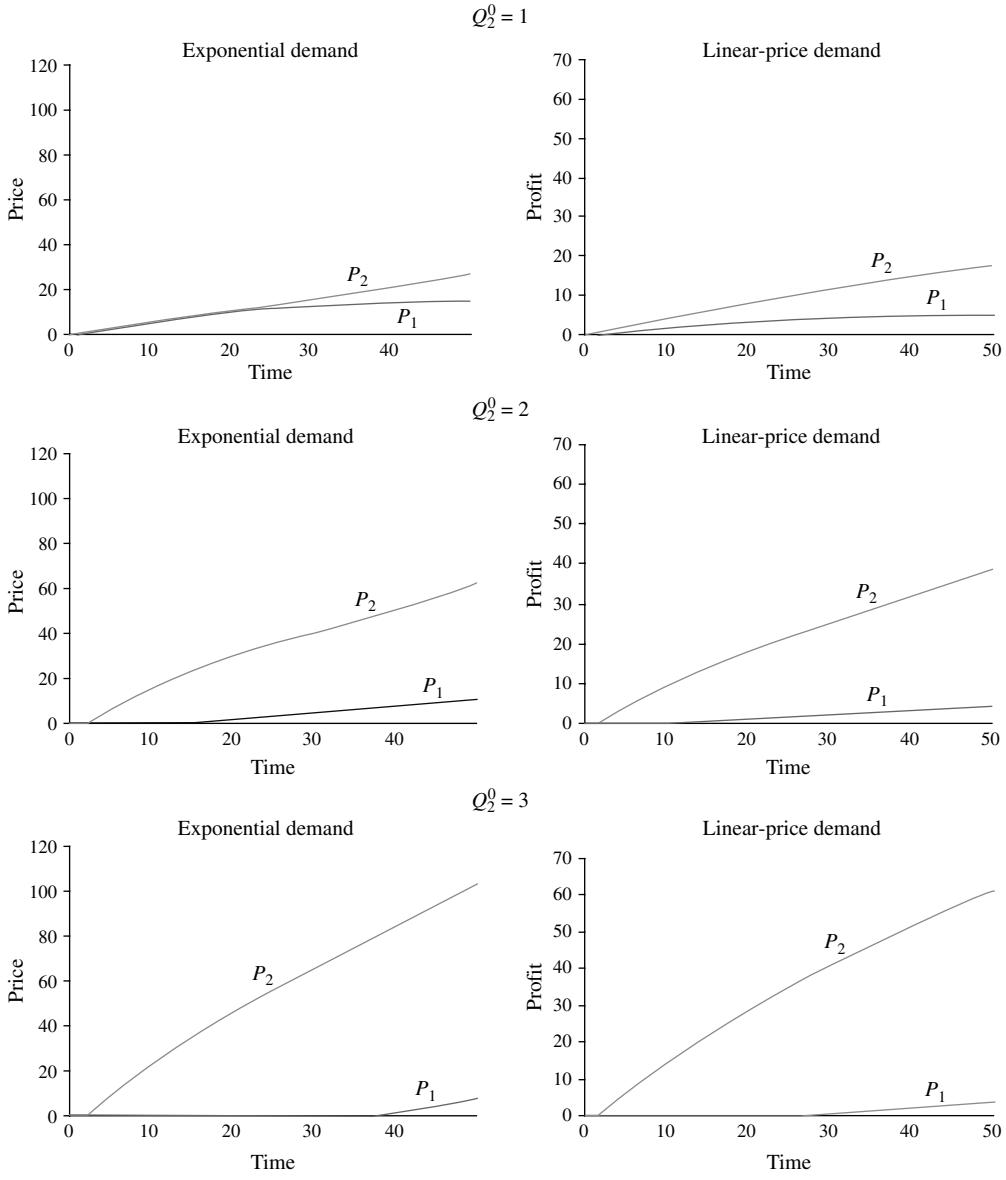


Figure 7 The Effects of Q_2^0 on P_1 and P_2 for Closed Source



2005, Haruvy et al. 2007). Compatibility and legal protection will become paramount when competition is considered. It should be added that the present framework need not be limited to open source strategies. A framework for joint optimization of hiring, quality, pricing, and network size is useful in any setting involving network development with multiple products. A natural extension is to freeware. Freeware strategies with a freeware and a commercial product (e.g., Adobe Acrobat Reader which is free, and Writer, which is not) are very similar to open source strategies, with the exception that users do not add directly to the code, though they do add to the quality through positive network externalities. The framework can also be extended to non-software domains with multiple products and network externalities.

References

Anderson, S. P. 1989. Socially optimal spatial pricing. *Regional Sci. Urban Econom.* **19**(1) 69–86.

Armstrong, M. 2006. Competition in two-sided markets. *RAND J. Econom.* **37**(3) 668–691.

Armstrong, M., J. Wright. 2007. Two-sided markets, competitive bottlenecks and exclusive contracts. *Econom. Theory* **32**(2) 353–380.

Carrillo, J., C. Gaimon. 2000. Improving manufacturing performance through process change and knowledge creation. *Management Sci.* **46**(2) 265–288.

Chintagunta, P. K., V. R. Rao. 1996. Pricing strategies in a dynamic duopoly: A differential game model. *Management Sci.* **42**(11) 1501–1514.

Cohen, M., J. Eliashberg, T.-H. Ho. 1996. New product development: The performance and time-to-market tradeoff. *Management Sci.* **42**(2) 173–186.

Cusumano, M., R. Selby. 1995. *Microsoft Secrets*. The Free Press, New York.

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.

Elmaghraby, W., P. Keskinocak. 2003. Dynamic pricing in the presence of inventory considerations: Research overview, current practices, and future directions. *Management Sci.* **49**(10) 1287–1307.

Fine, C. H. 1986. Quality improvement and learning in productive systems. *Management Sci.* **32**(10) 1301–1315.

Fine, C. H., L. Li. 1988. Technology choice, product life cycles, and flexible automation. *J. Manufacturing Oper. Management* **1**(4) 372–400.

Fink, M. 2003. *The Business and Economics of Linux and Open Source*. Prentice Hall, Upper Saddle River, NJ.

Gaimon, C. 1986. The optimal acquisition of new technology and its impact on dynamic pricing policies. B. Lev, ed. *Studies in Management Science and Systems*, Vol. 13. North Holland, New York, 187–206.

Gaimon, C. 1988. Simultaneous and dynamic price, production, inventory and capacity decisions. *Eur. J. Oper. Res.* **35**(3) 426–441.

Gaimon, C. 1989. Dynamic game results of the acquisition of new technology. *Oper. Res.* **37**(3) 410–425.

Gaimon, C. 1997. Planning information technology—Knowledge worker systems. *Management Sci.* **43**(9) 1308–1328.

Gallego, G., G. J. van Ryzin. 1994. Optimal dynamic pricing of inventories with stochastic demand over finite horizons. *Management Sci.* **40**(8) 999–1020.

Greenhut, J., M. L. Greenhut. 1977. Nonlinearity of delivered price schedules and predatory pricing. *Econometrica* **45**(8) 1871–1875.

Haruvy, E., A. Prasad. 1998. Optimal product strategies in the presence of network externalities. *Inform. Econom. Policy* **10**(4) 489–499.

Haruvy, E., A. Prasad. 2001. Optimal product strategies in the presence of network externalities: An evolutionary game theoretical approach. *J. Evolutionary Econom.* **11**(2) 231–248.

Haruvy, E., A. Prasad, S. Sethi. 2003. Harvesting altruism in open source software development. *J. Optim. Theory Appl.* **118**(2) 381–416.

Haruvy, E., A. Prasad, S. Sethi, R. Zhang. 2007. Competition with open source as a public good. *J. Indust. Management Optim.* **4**(1) 199–211.

Joglekar, N., A. Yassine, S. Eppinger, D. Whitney. 2001. Performance of coupled product development activities with a deadline. *Management Sci.* **47**(12) 1605–1620.

Kalish, S. 1983. Monopolist pricing with dynamic demand and production cost. *Marketing Sci.* **2**(2) 135–159.

Kalish, S., G. Lilien. 1983. Optimal price subsidy for accelerating the diffusion of innovation. *Marketing Sci.* **2**(4) 407–420.

Katz, M., C. Shapiro. 1985. Network externalities, competition and compatibility. *Amer. Econom. Rev.* **75**(3) 424–440.

Katz, M., C. Shapiro. 1986. Technology adoption in the presence of network externalities. *J. Political Econom.* **94**(4) 822–841.

Kouvelis, P., S. K. Mukhopadhyay. 1999. Modeling the design quality competition for durable products. *IIE Trans.* **31**(9) 865–880.

Kouvelis, P., D. Mallick, S. K. Mukhopadhyay. 1997. Dynamic interaction of price and design quality for products with convex production costs. *Production Oper. Management* **6**(4) 373–387.

Lee, D., H. Mendelson. 2008. Divide and conquer: Competing with free technology under network effects. *Production Oper. Management* **17**(1) 12–28.

Lerner, J., J. Tirole. 2005. The economics of technology sharing: Open source and beyond. *J. Econom. Perspectives* **19**(2) 99–120.

Mukhopadhyay, S. K., P. Kouvelis. 1997. A differential game theoretic model for duopolistic competition on design quality. *Oper. Res.* **45**(6) 886–893.

Muller, E., Y. Peles. 1988. The dynamic adjustment of optimal durability and quality. *Internat. J. Indust. Organ.* **6**(4) 499–507.

Nascimento, F., W. Vanhonacker. 1988. Optimal strategic pricing of reproducible consumer products. *Management Sci.* **34**(8) 921–937.

Norton, J. A., F. M. Bass. 1987. A diffusion theory model of adoption and substitution for successive generations of high-technology products. *Management Sci.* **33**(9) 1069–1086.

Parker, G., M. W. Van Alstyne. 2005. Two-sided network effects: A theory of information product design. *Management Sci.* **51**(10) 1494–1504.

Raymond, E. S. 1999. The magic cauldron. http://www.catb.org/esr/writings/magic-cauldron/magic-cauldron.html. Retrieved January 9, 2008.

Raymond, E. S. 2001. *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*. O'Reilly, Sebastopol, CA.

Rochet, J. C., J. Tirole. 2003. Platform competition in two-sided markets. *J. Eur. Econom. Assoc.* **1**(4) 990–1029.

Rogers, E. M. 1983. *Diffusion of Innovation*, 3rd ed. The Free Press, New York.

Schiff, A. 2002. The economics of open source software: A review of the early literature. *Rev. Network Econom.* **1**(1) 66–74.

Sethi, S. P., F. M. Bass. 2003. Optimal pricing in a hazard rate model of demand. *Optimal Control Appl. Methods* **24**(4) 183–196.

Sethi, S. P., G. L. Thompson. 2000. *Optimal Control Theory: Applications to Management Science and Economics*, 2nd ed. Kluwer Academic Publishers, Boston.

Shapiro, K., H. Varian. 1999. *Information Rules*. Harvard Business School Press, Boston.

Southwick, L., S. Zionts. 1974. An optimal control theory approach to the education-investment decision. *Oper. Res.* **22**(6) 1156–1174.

Tamai, T., Y. Torimitsu. 1992. Software lifetime and its evolution process over generations. *Proc. Internat. Conf. Software Maintenance, Orlando, FL*, IEEE Computer Society Press, Los Alamitos, CA, 63–69.

Zhang, R., E. Haruvy, A. Prasad, S. Sethi. 2005. Optimal firm contributions to open source software: Effects of competition, compatibility and user contributions. C. Deissenberg, R. Hartl, eds. *Optimal Control and Dynamic Games*: *Applications in Finance, Management Science, and Economics—Essays in Honor of Suresh Sethi* (*Advances for Computational Management Science Series*). Springer, New York, 197–214.