TECHNISCHE UNIVERSITÄT WIEN
Vienna University of Technology

Operations Research and Control Systems

Periodic Updates Relying on Open Source Software: An Impulse Approach

Andrea Seidl

Research Report 2012-11
December 2012

Operations Research and Control Systems
Institute of Mathematical Methods in Economics
Vienna University of Technology

Research Unit ORCOS
Argentinierstraße 8/E105-4,
1040 Vienna, Austria
E-mail: orcos@eos.tuwien.ac.at

Periodic Updates Relying on Open Source Software: An Impulse Approach*

Andrea Seidl

Abstract

This paper presents a way to locate a periodic solution in an impulse problem dealing with the question of when and how often to update a product using open source software. The Impulse Maximum Principle is applied to determine conditions for the corresponding boundary value problem. An initial solution can be found analytically for a simplified version of the model and continued numerically to comply with the original problem.

1 Introduction

Hardware specialized firms are often confronted with the problem that the sales of their product very much rely on the quality of the associated software. Particularly in the hotly contested smartphone market, for example, the features of the used operating system play an important role for customer's purchase decision. However, for many products software development is not done by the firm itself; see, e.g., Cromar (2010). Then the firm either licenses some proprietary software (e.g., Nokia currently uses Microsoft's Windows Phone 7 for some of its devices) or relies on open source software (e.g., Samsung, HTC, Motorola and others use Google's Android).

It is in the interest of the firm to keep the software of the product up-to-date, however, software updates are costly as they often require some product specific adaptations. This paper considers the problem of a firm with a product which relies on open source software which is continuously developed. Examples for such products include mobile phones, satnavs and digital video recorders which often use software derived from Linux. The firm is confronted with the questions of when and how often to update the product with the latest version of the open source software and whether and how much to contribute to the open source project.

The problem of a firm developing open source software with respect to investments and pricing of a complementary product is considered for example in Haruvy et al. (2008) within an optimal control framework. Caulkins et al. (2011) provide a multi-stage extension to study how the decision to make software open source is influenced by not considering the problem as a simple once and for all binary choice. Kort and Zaccour (2011) analyze the impact of competition on the decision to make software open source, also focusing on a business model where profits are made by the sales of a complementary product.

Haruvy et al. (2005, 2008) study the problem of a firm selling software consisting of open source and proprietary components within a differential game.

Economides and Katsamakas (2006) consider a two-stage game between a software platform provider and of a firm selling a complementary product, however, they do not look at change of quality of the products over time and therefore do not consider the problem of software updates.

*This research was supported by the Austrian Science Fund (FWF) under Grant P21410-G16. I wish to thank Anton Belyakov, Mohammed Chahim, Gustav Feichtinger, Dieter Grass and Peter Kort for discussion and helpful comments.

An impulse approach is used in Mukherji et al. (2006) in a technology adoption model dealing with the question of when to invest in information technology upgrades. Chahim et al. (2012a) apply the Impulse Maximum Principle as derived in Chahim et al. (2012b) to study the problem of a firm that has the option to invest in product innovation at an optimally determined time.

The purpose of this paper is to look at how optimal long run solutions of an impulse problem may look like. The particular focus lies on how to determine a periodic solution within this framework. By simplifying the model one can formulate a boundary value problem that can be solved analytically and continued numerically to comply with the original problem. Further, the paper provides some basic economic interpretation of the model and the results. In particular, we will see how the pricing strategy and the investments into software quality depend on the timing of software updates along the periodic solution. The next section presents the model and provides the necessary conditions for optimality, Section 3 describes the periodic solution and Section 4 concludes.

2 An Impulse Approach Related to Software Updates

2.1 The Model

The firm is a monopolist and tries to maximize its profits arising by the sales of a product having hardware and software components. Thus, we have

$$\max_{p(\cdot),v(\cdot),\tau,N} \int_0^\infty e^{-rt}(p(t)q(K_2(t),p(t)) - \beta v(t))dt - \sum_{i=1}^{N} c(w),$$

where the demand for the product is given by

$$q(K_2(t),p(t)) = K_2(t) + \alpha - \varphi p(t).$$

Parameter α describes the impact of the quality of the components independent of the used open source software on demand and is for simplicity assumed to be constant. An infinite time horizon is chosen as we do not consider a specific version of a product and its life cycle, but the evolution of the product as a whole. The software related quality is given by state variable $K_2(t)$ and is measured by the demand it generates if the price of the product is zero. φ relates to the impact of price $p(t)$ on demand. Control variable $v(t)$ denotes the efforts of the firm in order to contribute to the open source project leading to costs described by parameter β. Updating the software of the product at time τ_i leads to adaption costs described by function $c(w) = \pi w^\gamma$ with $\gamma > 1$, where w denotes the contribution to the quality when adapting the software for the firm's purposes. In this simple model w is assumed to be constant. There are control constraints, i.e.

$$v(t) \geq 0, \qquad p(t) \geq 0, \qquad q(K_2(t),p(t)) > 0,$$

but it is easy to show that they cannot become active, therefore they are not considered in more detail in the subsequent. The quality of the open source software $K_1(t)$ evolves according to

$$\dot{K}_1(t) = h(v(t)) + g(D(K_1(t))) - \delta_1 K_1(t), \tag{1}$$

where function $h(v(t))$ describes the firm's contribution to the open source project. $g(K_1(t))$ captures the contribution of the open source community, which depends on the demand for the product and, consequently, on the quality of the open source project. Like in Caulkins et al.

(2011), it is assumed that $h(v(t)) = av(t)^b$ and $g(D(K_1(t))) = mK_1(t)^n$, with $b, n < 1$. δ_1 denotes the rate of technological obsolescence of the open source software. Note that the open source software itself cannot be sold by the firm and consequently $K_1(t)$ does not enter the objective function directly.

The state equation of the software related quality $K_2(t)$ of the full product then is

$$\dot{K}_2(t) = -\delta_2 K_2(t) \qquad \text{for } t \notin \{0, \tau_1, ..., \tau_N\}. \tag{2}$$

Parameter δ_2 describes the rate of technological obsolescence of the product related to the used software. It is assumed that the open source software is not exclusively developed for the specific product, therefore $\delta_1 > \delta_2$ would mean that technological obsolescence affects demand for the open software more severely than the demand for the complete product.

At time τ_i the product is updated with a newer, adapted version of the software. Thus, we have

$$K_2(\tau_i^+) = \epsilon K_1(\tau_i^-) + w, \tag{3}$$

where ϵ relates to the impact of the quality of the open source software on the quality of the product consisting of hardware and software components. It should be mentioned that an open source software without copy-left licensing terms is considered, i.e. the firm is allowed to use the software for its proprietary product. Note that it is assumed that every time the product is updated, it is done with an entirely new adaptation of the open source software, i.e. the quality after the jump does not depend on the quality of the product before the jump. This makes sense when adaptations cannot be reused, e.g. due to compatibility issues. The quality of the open source software does not change at the jump point.

In the subsequent time argument t will be omitted unless necessary for understanding.

2.2 Necessary Conditions

The Hamiltonian is

$$\mathcal{H} = p(K_2 + \alpha - \varphi p) + \lambda_1(av^b + mK_1^n - \delta_1 K_1) + \lambda_2(-\delta_2 K_2).$$

In order to be able to apply the Impulse Maximum Principle, see Chahim et al. (2012b), we have to rewrite (3) as

$$K_2(\tau_i^+) - K_2(\tau_i^-) = \epsilon K_1(\tau_i^-) + w - K_2(\tau_i^-).$$

The Impulse-Hamiltonian is

$$\mathcal{IH} = -\pi w^\gamma + \lambda_2(\tau_i^+)(\epsilon K_1(\tau_i^-) + w - K_2(\tau_i^-)).$$

It can be found that

$$v = \left(\frac{ab\lambda_1}{\beta}\right)^{\frac{1}{1-b}}, \qquad p = \frac{K_2 + \alpha}{2\varphi}. \tag{4}$$

The costate equations are

$$\dot{\lambda}_1 = (r + \delta_1 - mnK_1^{n-1})\lambda_1, \tag{5}$$

$$\dot{\lambda}_2 = (r + \delta_2)\lambda_2 - p. \tag{6}$$

At the impulse point it has to hold that

$$\lambda_1(\tau_i^+) - \lambda_1(\tau_i^-) = -\epsilon\lambda_2(\tau_i^+), \tag{7}$$

$$\lambda_2(\tau_i^-) = 0, \tag{8}$$

$$\mathcal{H}(\tau_i^+) - \mathcal{H}(\tau_i^-) - r\pi w^\gamma = 0. \tag{9}$$

2.3 Optimal Long Run Solution

Steady states and limit cycles are candidates for solutions fulfilling the necessary transversality conditions, see, e.g., Blaquière (1985); Feichtinger and Hartl (1986).

In the present model we can find a steady state at

$$(\hat{K}_1, \hat{K}_2, \hat{\lambda}_1, \hat{\lambda}_2) = \left(0, 0, 0, \frac{\alpha}{2\varphi(r+\delta_2)}\right).$$

Looking, however, at (1) we can see that this steady state cannot be reached for any $K_1(0) > 0$.

There is a second steady state at

$$(\hat{K}_1, \hat{K}_2, \hat{\lambda}_1, \hat{\lambda}_2) = \left(\left(\frac{m}{\delta_1}\right)^{\frac{1}{1-n}}, 0, 0, \frac{\alpha}{2\varphi(r+\delta_2)}\right).$$

Evaluating the Jacobian at this steady state we find two positive eigenvalues ($\xi_1 = \delta_2 + r$, $\xi_2 = (1-n)\delta_1 + r$) and two negative eigenvalues ($\xi_1 = -\delta_2$, $\xi_2 = (n-1)\delta_1$) implying that this steady state is a saddle point with a two-dimensional stable manifold. Thus, trajectories leading to this steady state cannot be excluded as candidates for an optimal solution. Yet, we can see that demand for the product is only caused by aspects of the product not related to the software at this long run solution. Of course such a solution can make sense when software updates are too expensive in the long run.

Another possibility for an optimal long run solution is to approach a limit cycle. However, locating a periodic solution is not as straightforward as determining a steady state. The next section considers how it is possible to find a periodic solution for this particular model.

3 Finding a Periodic Solution

The essential property of a periodic solution $\Gamma(\cdot)$ is that for $x(t) = (K_1(t), K_2(t), \lambda_1(t), \lambda_2(t))'$ and $x(0) \in \Gamma(\cdot)$ it has to hold that

$$x(t) = x(t+T) \qquad \text{and} \qquad x(t^+) = x\left((t+T)^+\right),$$

where the period of the cycle T is

$$T = \tau_{i+1} - \tau_i \qquad \text{for all } i \geq 1.$$

Considering (3) and (7)-(9), to find a periodic solution we have to solve the *boundary value problem* (BVP) (1)-(2), (5)-(6) subject to the boundary conditions

$$\begin{aligned}
K_1(0^+) &= K_1(T^-), \qquad K_2(0^+) = \epsilon K_1(T^-) + w, && (10)\\
\lambda_1(0^+) &= \lambda_1(T^-) - \epsilon\lambda_2(0^+), && (11)\\
\lambda_2(T^-) &= 0, && (12)\\
\mathcal{H}(0^+) &= \mathcal{H}(T^-) + r\pi w^\gamma, && (13)
\end{aligned}$$

on the time horizon $[0, T]$, where it is assumed that there is only one jump which occurs at the initial/final point of the cycle. This cannot be solved analytically, therefore we have to resort to numerical methods. However, numerical methods to solve BVPs, like Matlab's `bvp4c`, require an initial solution. In the next section a simplified boundary value problem is solved, which's solution can be used for a continuation algorithm to numerically find a solution of the full problem.

3.1 Finding an Initial Solution in a Simplified Version of the Model

Let us assume that parameter $a = 0$. In terms of interpretation this means that no contribution of the firm is admitted to the open source project. Consequently, the firm does not invest in any contribution, i.e., $v = 0$. But this means, considering (1) that in the long run the quality of the open source software is constant, i.e.,

$$K_1(t) = \hat{K}_1 = \left(\frac{m}{\delta_1}\right)^{\frac{1}{1-n}}. \tag{14}$$

Solving (2) s.t. (10) gives

$$K_2(t) = \left(\epsilon \hat{K}_1 + w\right) e^{-\delta_2 t}. \tag{15}$$

Considering (4), (6) and (12), and denoting $K_{20} = K_2(0^+)$, $\varrho = r + \delta_2$ and $\theta = r + 2\delta_2$, it can be found that

$$\lambda_2(t) = \frac{1}{2\theta\varrho\varphi} e^{\varrho t} \left(\varrho K_{20}(e^{-\theta t} - e^{-\theta T}) + \alpha(e^{-\varrho t} - e^{-\varrho T})\theta\right) \tag{16}$$

Proposition. A periodic solution is only possible if $r + \delta_1 > n\delta_1$.

Proof. Boundary condition (11) implies that $\lambda_1(T^-) > \lambda_1(0^+)$. Inserting (14) into (5), it can be seen that $\dot{\lambda}_1(t) > 0$ iff $r + \delta_1 > n\delta_1$ for $t \neq T$. ■

Since it is assumed that $n < 1$ this condition is always fulfilled. Also note that the mere existence of a periodic solution does not mean that it is optimal or even possible to approach it.

It makes sense that the costate related to the open source software's quality increases over time (except at the jump point). This relates to the fact that a higher quality of the open source software is more valuable close to the jump point.

By using (5), (11) and (16), and defining function $\sigma(t) = e^{\frac{t}{\hat{K}_1}(-mn\hat{K}_1^n + (r+\delta_1)\hat{K}_1)}$ the first costate can be determined as

$$\begin{aligned} \lambda_1(t) \;=\; & \frac{\sigma(t)\epsilon}{4(1-\sigma(T))\varphi\varrho(\frac{r}{2}+\delta_2)}(\varrho K_{20} e^{-\theta T} \\ & +2\alpha(\frac{r}{2}+\delta_2)e^{-\varrho T} - (K_{20}+2\alpha)\delta_2 - r(K_{20}+\alpha)) \end{aligned} \tag{17}$$

By inserting (14)-(17) into (13) time period T can be determined, however, this has to be done numerically.

Proposition. A periodic solution with more than one jump during one period of the cycle is not possible.

Proof. Let us assume that there are two jumps at $t = \tau_1$ and at $t = T = \tau_1 + \tau_2$.

It can be easily shown that if $\tau_1 = \tau_2$ the actual period of the cycle is not T, but $\tau_1 = \tau_2 = \frac{T}{2}$.

Let us now assume that $\tau_1 > \tau_2$. The problem is autonomous and time-consistent, thus let us split the cycle into two parts, which we distinguish by superscripts $j = 1, 2$. It is assumed that jumps occur at the initial point of each subsolution. Let us now consider these solutions on the intervals $[0, \tau_1]$ and $[0, \tau_2]$, where we have due to (3) the boundary conditions $K_2^1(0^+) = K_2^2(\tau_2^+) = K_{20}$ and $K_2^2(0^+) = K_2^1(\tau_1^+) = K_{20}$. Considering (2) and (3), $\tau_1 > \tau_2$ means that $K_2^1(\tau_1^-) < K_2^2(\tau_2^-)$. Consequently, $\mathcal{H}(\tau_1^-) < \mathcal{H}(\tau_2^-)$. Matching condition (9) then implies that $\mathcal{H}(\tau_1^+) < \mathcal{H}(\tau_2^+)$. Since $K_2^1(\tau_1^+) = K_2^2(\tau_2^+) = K_{20}$ this condition can only be fulfilled if $\lambda_2^1(\tau_1^+) > \lambda_2^2(\tau_2^+)$. Since we have a cycle this means it has to hold that $\lambda_2^2(0^+) > \lambda_2^1(0^+)$. Solving our problem now for λ_2, we find that $\lambda_2^2(0^+) > \lambda_2^1(0^+)$ only if $1 - e^{-\theta\tau_2} > 1 - e^{-\theta\tau_1}$ or $1 - e^{-\varrho\tau_2} > 1 - e^{-\varrho\tau_1}$. As these conditions cannot be fulfilled since $\tau_1 > \tau_2$, there is a

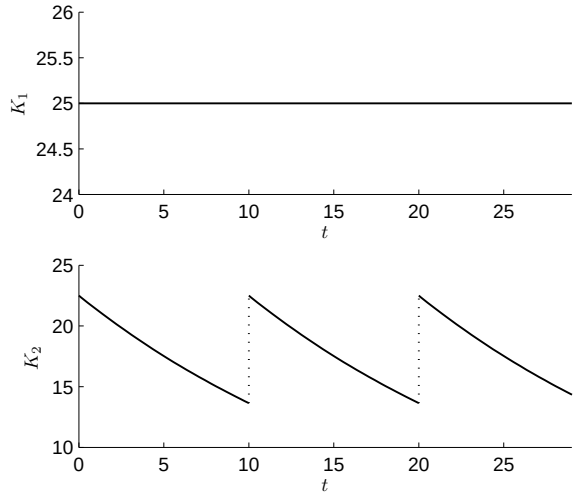

Figure 1: Time path for the state variables on the periodic solution for $a = 0$.

contradiction finishing the proof. The idea of this proof can be applied to three and more jumps. ■

Table 1: Parameter values used for the numerical calculations

r	α	φ	δ_1	δ_2	a	b
0.1	1	5	0.05	0.05	0	0.5
m	n	ϵ	w	π	γ	
0.25	0.5	0.5	10	0.536	2	

Fig. 1 displays the periodic solution over time for the parameters displayed in Table 1. (Note that here in fact parameter π was chosen such that $T = 10$.) Obviously due to the used assumptions, the quality of the open source software remains constant and the software related quality of the complete product decreases over time except at jump points.

Fig. 2 reveals the corresponding optimal controls. While it is not optimal to contribute to the open source project, i.e., $v = 0$, the price of the complete product decreases over time in order to compensate the decrease in demand due to technological obsolescence. When updating the software the price can again be increased.

3.2 Solution of the Original Boundary Value Problem

Applying now a simple continuation algorithm as described in Grass et al. (2008), the original boundary value problem can be solved numerically by using the solution obtained in the previous section as initial guess and gradually increasing parameter a.

Fig. 3 reveals the time path for a solution starting on the cycle, Fig. 4 reveals the corresponding controls. It can be seen now that the quality of the open source software and the firm's contributions to it do not remain constant anymore in the long run. Immediately after a software update the firm's incentive to contribute to open source software development becomes rather small, but increases continuously. The reason for this is that once the product is updated with the latest version of the software, a firm does not care as much about the quality

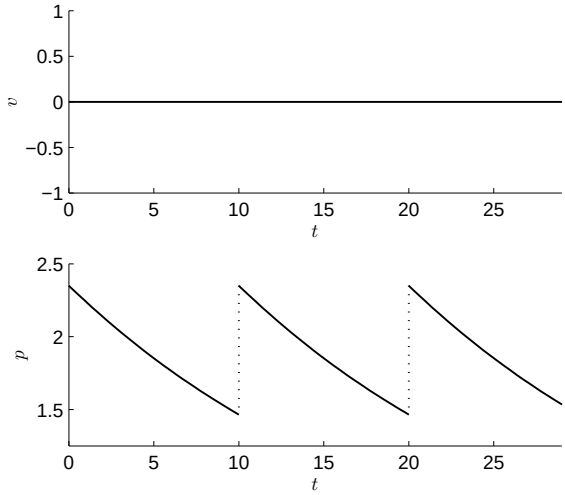

Figure 2: Time path for the control variables on the periodic solution for $a = 0$.

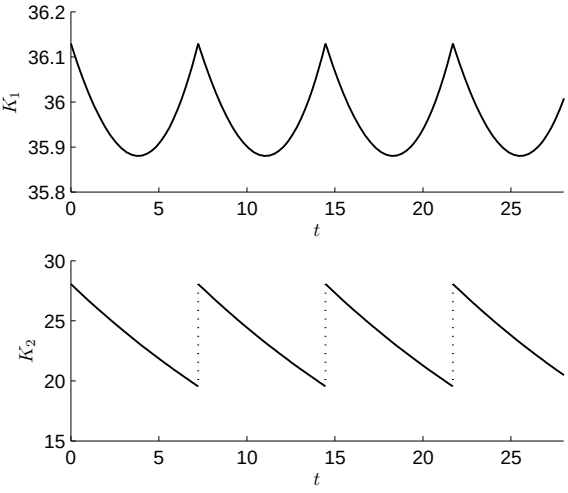

Figure 3: Time path for the state variables on the periodic solution for $a = 1$.

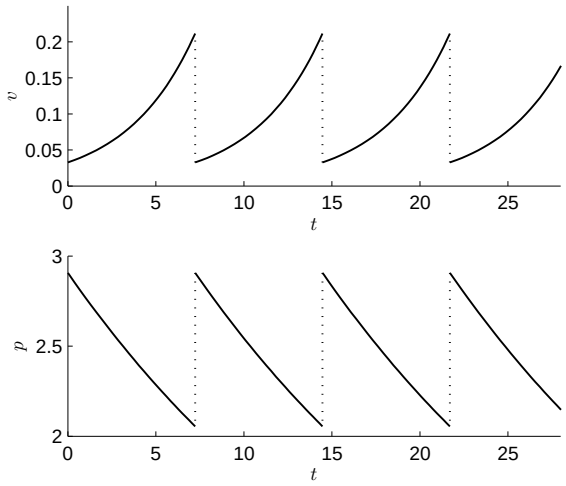

Figure 4: Time path for the control variables on the periodic solution for $a = 1$.

of the open source software as close to the jump. Resultingly, after the jump the quality of the software suffers given the lack of contribution from the firm. Thus, the quality of the open source software decreases, but once the firms contributions have reached a higher level it starts to recover again.

The strategy with respect to pricing is the same as in the simplified version of the model: A jump in the software related quality of the product allows the firm to increase the price, but due to technological obsolescence the price is lowered over time to strengthen demand.

The quality of the open source project, and, consequently, the software related quality of the complete product profit from the firm's efforts to participate in the software development process when the firm's contribution are admitted. Therefore, the firm can charge a higher price for the complete product and afford a higher update frequency than if it is not involved in the software development. Note, however, higher costs for software development would lower the firm's efforts for contributions.

4 Conclusion

The present paper is a starting point for a closer consideration of the questions related to products whose sales depend on the updates of the underlying software. The necessary conditions for an optimal solution are derived and steady states determined. In a simplified version of the model a periodic solution can be found analytically for the most part by solving a boundary value problem. The resulting solution can be used for a numerical continuation algorithm by which a solution of the original problem can be found.

The next step in the analysis of the problem would be to consider how solution paths with initial values not lying on the periodic solution look like and to do a profound sensitivity analysis with respect to interesting parameters.

In terms of interpretation it is optimal for a firm to raise prices for the complete product only in compliance with software updates. Otherwise it has to decrease prices to strengthen demand which suffers under technological obsolescence. At the moment when software is updated a firm looses some incentive to keep up its efforts to contribute to the open source software. However, as the firm wants to profit from a competitive software, its efforts to support the development

of the open source software increase over time.

The model presented here can be extended in several ways. It can be adapted to deal with the problem of a firm relying on proprietary software developed by another firm. Of course, it is also interesting how competition affects the firm's motivation to contribute to the open source software. Further, it would be interesting to look at what happens if the quality of the underlying hardware is not constant but is subject to continuous developments.

References

Blaquière, A. (1985). Impulsive optimal control with finite or infinite time horizon. *Journal of Optimization Theory and Applications*, 46(4):431–439.

Caulkins, J., Feichtinger, G., Grass, D., Hartl, R., Kort, P., and Seidl, A. (2011). When to make proprietary software open source. ORCOS Research Report 2011-07, Vienna University of Technology. In Submission.

Chahim, M., Grass, D., Hartl, R., and Kort, P. (2012a). Product innovation with lumpy investment. In progress.

Chahim, M., Hartl, R., and Kort, P. (2012b). A tutorial on the deterministic Impulse Control Maximum Principle: Necessary and sufficient optimality conditions. *European Journal of Operations Research*, 219(1):18–26.

Cromar, S. (2010). Smartphones in the U.S.: Market analysis. available at https://www.ideals.illinois.edu/handle/2142/18484 (last accesed Feb, 23^{rd}, 2012).

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

Feichtinger, G. and Hartl, R. F. (1986). *Optimale Kontrolle ökonomischer Prozesse: Anwendungen des Maximumprinzips in den Wirtschaftswissenschaften*. Walter de Gruyter, Berlin.

Grass, D., Caulkins, J. P., Feichtinger, G., Tragler, G., and Behrens, D. A. (2008). *Optimal Control of Nonlinear Processes: With Applications in Drugs, Corruption and Terror*. Springer, Heidelberg.

Haruvy, E., Prasad, A., Sethi, S., and Zhang, R. (2005). Optimal firm contributions to open source software: Effects of competition, compatibility and user contributions. In Deissenberg, C. and Hartl, R., editors, *Optimal Control and Dynamic Games: Applications in Finance, Management Science, and Economics*, pages 197–212. Springer.

Haruvy, E., Sethi, S. P., and Zhou, J. (2008). Open source development with a commercial complementary product or service. *Production and Operations Management*, 17(1):29–43.

Kort, P. M. and Zaccour, G. (2011). When should a firm open its source code: a strategic analysis. *Production and Operations Management*, 20(6):877–888.

Mukherji, N., Rajagopalan, B., and Tanniru, M. (2006). A decision support model for optimal timing of investments in information technology upgrades. *Decision Support Systems*, 42(3):1684–1696.