Deciding On Software Pricing And Openness Under Competition

Hazhir Rahmandad, hazhir@vt.edu
Thanujan Ratnarajah, thanujan@vt.edu

Abstract
The success of many open source applications has motivated commercial firms to explore how they can benefit from opening their software platforms in hope of getting free high quality contributors and more complementary products. Yet the openness decision is tightly coupled with the pricing of the software (e.g. openness limits the price that can be charged) and the reinforcing feedback loops of network effects and complementary products. In this paper we explore how there interconnections impact the optimum pricing and openness decision for two firms in competition. Reinforcing loops increase the value of early market lead and put pressure on the competing firms to seek such advantage. We show that the competitive equilibrium under strong reinforcing loops calls for highly open software products with deep early discounts, which may significantly compromise the profitability of the players in the market. Proprietary platforms and higher prices are favored in the absence of these loops.

Keywords: software, network effects, complementary goods, open source, Nash equilibrium

1- Introduction and Problem Definition

Software development is a major part of economy in many advanced and developing countries. Due to innovative nature of this business and competitive pressures new development processes and organizational structures routinely emerge in this market. Opening the code and licensing of a software product has attracted significant attention over the last few years after the success of many open source projects such as Linux, Netscape, and Apache. Open source software has proven a strong competitor for many traditional software businesses. Once only available in marginal markets and simple products, open source software is increasingly competing with commercially developed products in complex applications and where high quality and reliability are required. The result is a significant threat for the traditional software business model which relies heavily on the sales revenue of the software.

Another major trend in the recent years is strategizing to benefit from network externalities of software products and the impact of complementary products and standardization on market performance of a new product. Many firms therefore have moved towards get-big-fast strategies that by reducing initial price help them take market share early in the game to enable future success (Oliva, Sterman et al. 2003). Moreover, these two trends interact in forming business strategy for a firm: for example openness of a new software impacts its attractiveness for developers of complementary products, the price that can be charged on it, and the costs of development.

For-profit software companies need to find avenues to adapt their own software business models to the new challenges and potential opportunities created by open source movement on the one hand and the strong reinforcing loops active in software business on the other. We examine how the processes of development, diffusion, and revenue generation of a software product evolve in interaction with strategic decisions regarding the business model used by the software company. The research develops a quantitative model that can help software companies assess tradeoffs involved in selecting different business models and thus allow for better strategic decisions.

1-1- Openness: A continuum of licensing models

Open source software development has been characterized by highly motivated communities of developers who on their own time and without direct financial incentives work on developing software products. These communities openly share the code of the software and use licensing measures that keep this code in the public domain. Such development model can lead to significant cost savings compared to traditional software development model both due to sharing of modules of common code and savings on the cost of developers. On the other hand status, learning, and career advancement incentives, among others, can explain why open source contributors stay engaged in these communities (Lakhani and von Hippel 2003; Lerner and Tirole 2005). High quality and reliability of many open source products has further increased their competitiveness in the market against traditional proprietary mode of software development and distribution.

A second benefit of opening the source code to a product is how that move encourages other firms and developers to build complementary software that adds value to the central software. The popular social networking website Facebook (over 175 million users, and growing (Facebook 2009)) provides a good example. In 2007 it opened its platform to third-party applications that merge with the current system. In less than 6 months over 8000 third party applications were developed and added to the system. This number has grown to over 52,000 applications by early 2009, and growing with the rate of 140 applications per day (Facebook 2009).

The facebook example highlights the continuity of openness. While in its pure form open source software is free and the code is publicly available, there are many different licensing methods that can put a software product on a continuum of purely open to completely proprietary. For example the Facebook system is not open source in that outsiders can not change the structure of Facebook program. Yet it also provides integration points and access to users for third party applications, thus creating some level of openness. More generally firms can keep some parts of the code open and benefit from open source development while keeping other parts propriety (Nilendu and Madanmohan 2002). Some of the most common licensing arrangements that define different points on the openness spectrum include (Wu and Lin 2001):

GNU GPL: This particular license emphasizes on the source code of the software always being open to the community. If a software project is licensed under this model the community involved in this project will have the rights to make improvements without the permission of the owner. But any improved work or derived work from the source code should be made public. This license prevents open source projects being mixed with proprietary licenses. GPL maintains a pure open source format.

Mozilla Public License (MPL): MPL is considered to be a weaker definition of GPL in terms of keeping source open. Any open source work should always be licensed freely. But the open source work could be combined with other licenses (including proprietary license) as long as the open source portion of the software stays open to the public. The organization need not publish the other non-open source work existing within the software, to the public.

BSD Licenses: BSD licenses allow both improved and derived open source work to be licensed under proprietary licenses. It has no limiting constraints such as GPL. Open source work could be easily combined with proprietary work.

Proprietary Licenses: Licensee is permitted to use the software under the conditions mentioned in the license created by the owner of the software. Using software (under this license) for improvement activities or derived work is prohibited. It could only be done with the permission of the owner of the software.

While the more open modes of development have many benefits in terms of development cost and encouraging complementary products, they pose a significant challenge in terms of commercial viability. The strictest forms of open source software are by definition free: if anybody can download and alter the code, you can not charge for that code. The lower levels of openness may allow for charging for software, yet they usually bring down the price to accommodate community of developers and complementary products. As a result, direct sales revenue is negatively impacted by openness of a software product, even though maintenance and customization of installed software, and advertising revenue in case of online software, could still provide some stream of revenue. This poses an important question for the profit oriented developers of a new software product: given all the tradeoffs involved what level of openness should they choose to maximize their long-term profit? Given the legal and technical ramifications of openness question, once settled, it is not easy to change the level of openness for a product.

1-2- Network effects and pricing in competition

Over the past two decades management scholars and economists have paid increasing attention to the significance of different reinforcing loops at work in the operation of firms. Network effects (Katz and Shapiro 1992), word of mouth (Dodson and Muller 1978), learning curves (Argote and Epple 1990), and economies of scale and scope (Panzar and Willig 1981) are a few of such reinforcing loops. The common denominator of these mechanisms is that as the size of the company or its market share grow, its unit cost for further growth goes down. For example more fax machines make the next unit of fax machine more attractive (network effect), satisfied customers lower the cost of attracting new customers (word of mouth), accumulating production leads to more efficient production processes (learning curves), and larger size reduces fixed cost per unit of product (economies of scale). As a result the larger firms find themselves in a better position in competition, and can derive the competitors out in "winner take all" dynamics (Frank and Cook 1995). When these loops are active, early success breads further advantage to the firm and thus an aggressive policy of expansion is prescribed for firms competing in markets with strong reinforcing loops (Arthur 1989).

Companies typically can gain early advantage by a combination of entry timing (those who enter the market earlier get a head start), early discounts, and joining in networks of complementary goods to increase the value of their product to early adopters (e.g. one of the reasons why VHS won against Beta max in the video market (Sterman 2000)). Previous research has explored the optimum pricing policy for a firm benefiting from learning curves (Spence 1981; Cabral and Riordan 1994) and has shown that early discounts can help firms succeed in competition when learning curves benefits can be kept inside the firm. Therefore when reinforcing loops are important in a market, pricing decisions should be made with an eye on these dynamics, and typically efficient prices change over time depending on market conditions.

Many software products are exposed to multiple reinforcing dynamics including word of mouth (Dellarocas 2003) (e.g. for viral marketing), network externalities (e.g. computer games are more attractive when many of your social network are playing them), complementary software development (e.g. Microsoft Windows platform is largely attractive because most mass market software firms first develop their product for this platform), and economies of scale (e.g. most of the cost of software development is fixed costs and production and distribution costs are often negligible). Therefore a dynamic pricing policy that considers these feedback effects seems a requirement for software pricing. Moreover, finding optimal pricing policy for new software products is linked to the openness question discussed above: openness decision sets a limit on the prices that can be charged on a product (Hawkins 2004). Thus the two decisions should be considered simultaneously: what pricing policy and what openness level should a new firm adopt?

Another complexity in determining optimum pricing policy relates to the competitive nature of the these decisions. Firms are not making their pricing decisions independently, rather, their decision directly depends on the decisions of their competitors. For example a reduction in price can lead the competitor to reduce theirs, negating the expected benefit of the initial move. Therefore a game theoretic framework should be employed to find out the Nash equilibriums in such competitions. In such equilibrium neither player can improve their payoff by changing their pricing (and other) policies. Finding Nash equilibrium is however typically hard analytically for all but simplest discrete time models.

The goal of this paper is to advance a solution to the challenge faced by software firms using a generic dynamic model of software development, sales, and complementary products in competition between two firms. We model how firms compete using price and openness to capture a pool of customers and how their success in getting customers impacts their profits. We then introduce a numerical method for solving the game theoretic equilibrium problem for these firms and analyze the structure of this solution as a function of alternative market and technology characteristics.

2- Modeling dynamics of software openness and pricing

Our modeling work draws on data from two case studies and the relevant literature. One proprietary study by the first author explored the costs and benefits of different openness decisions that could be pursued by a large software company in positioning one of its products. Given the proprietary nature of this case we only used it to inform the formulation of development and market share, however, the parameter values reported here differ from the case. Anther case using data from the competition of Linux and Windows operating systems informed the characteristics of reinforcing loops active in software development. That case is documented elsewhere (Ratnarajah 2008). Both these cases informed the structural features of the model in terms of software life cycle, factors determining the attractiveness, and the important feedback processes. They also helped us determine reasonable ranges for specific parameters. However, the model reported in this paper does not attempt to reflect any specific product. Capturing some of the most critical feedback mechanisms involved in determining pricing and openness policies, this model is used to find general patterns of optimal pricing and openness under alternative market and technology settings and can be tailored to specific cases as needed.

The profitability of a software business depends on its costs and revenues. Figure 1 outlines the major interactions between the market and the software organizations and how these shape the costs and revenues in our model. Note that similar structures are replicated for both firms (though subscripts in Vensim™ simulation environment). The revenue of a software company relies on the market share, which determines the product sales, and the installed base that controls service revenue. Revenue directly and indirectly depends on the openness and pricing decisions. Higher prices increase revenue directly and openness contributes to revenue indirectly by increasing the market share through increasing complementary software and thus customer utility. Openness also reduces the costs by replacing paid employees with open source community of developers. Multiple reinforcing loops that depend on the installed base can further increase the services revenue and the attractiveness of strategies that benefit from openness of the software.

On the other hand, the openness of the platform requires a company to allow other parties access to the result of their development activities. This sharing limits the product price that can be charged and thus negatively impacts the revenue. Multiple feedback processes result from these interactions and lead to tradeoffs that influence optimum pricing and openness decisions.

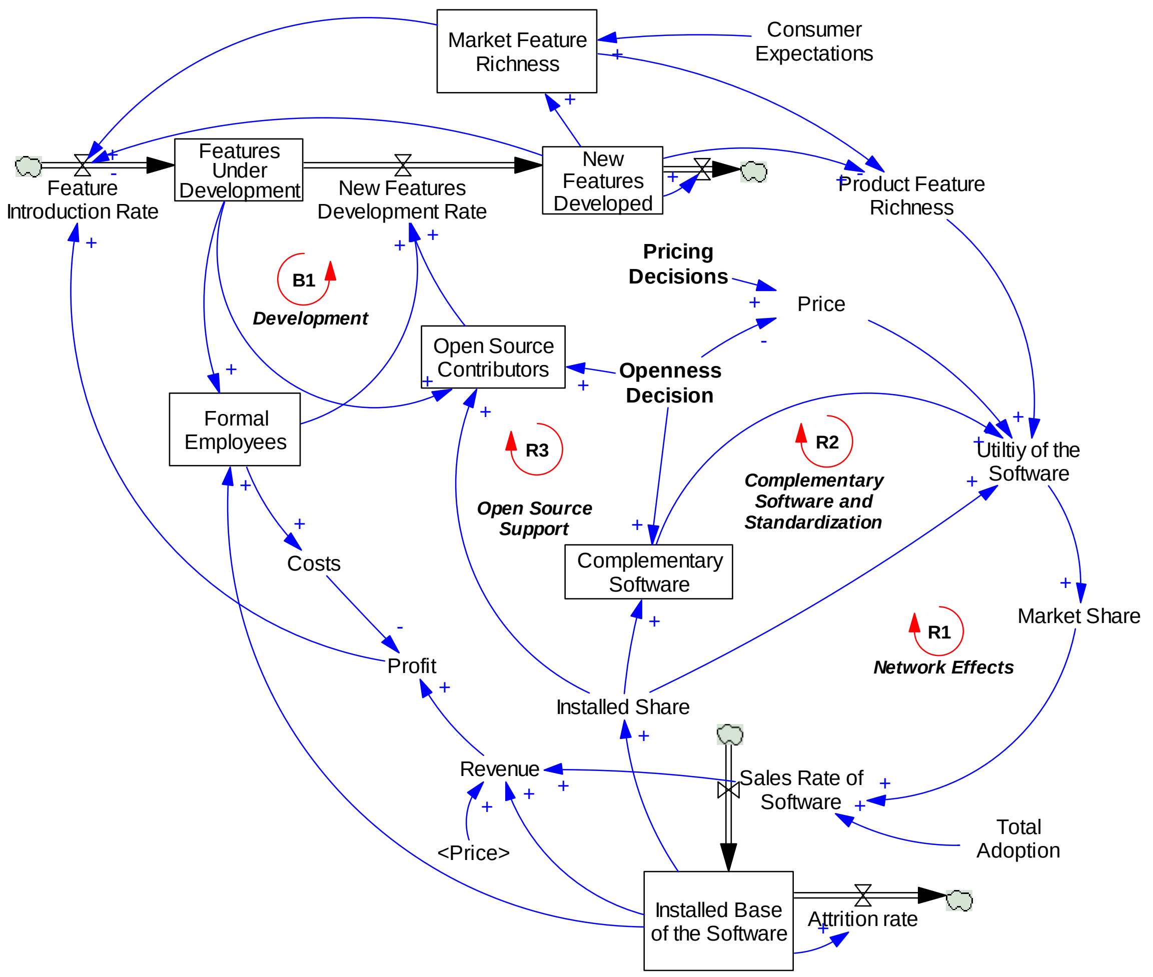


Figure 1- The overview of the feedback loops under analysis.

We consider the software development companies to decide on developing new features (Feature Introduction Rate) based on benchmarking their current features (New Features Developed) against capabilities of average product in the market (Market Feature Richness) and trying to move ahead of the market. The market feature richness partially depends on the features provided by all the players in the market, and partially is exogenous based on initial consumer expectations. After an initial investment period, introducing new features also depends on profitability of the software: if profitability is low, investment dries up. The relative feature richness of the product impacts its utility to consumers.

Both "Open Source Contributors" and "Formal Employees" can contribute to the development of the software. The numbers of these contributors partially depend on the number of "Features Under Development". Number of open source developers also depends on the openness of the software platform and its current traction in the market. The firm adjusts its formal employees to finish the features under development in the planned time, given the current level of open source contributors resulting in the balancing loop B1:Development. A software product with a large pool of pen source contributors requires fewer formal employees and thus lower Costs.

Sales of software (Sales Rate of Software: SS) depends on the "Total Adoption" rate (A), which we assume to be exogenous and determined by a typical product life cycle model (i.e. a bell-curve over time), and "Market Share" (MS) for each player in the market. Market share in turn depends on "Utility of the Software" (U) through the classical Logit model:

$$MS_i = \frac{e^{U_i}}{\sum e^{U_i}} \qquad (1)$$

Utility of the software is determined based on an additive function of "Product Feature Richness" (*F*), "Price" (*P*), "Complementary Software Share" (*C*), and "Installed Share" (S) for the product:

$$Ui = \alpha_F F^{\beta_F} - \alpha_P . P_i^{\beta_P} + \alpha_C . C_i^{\beta_C} + \alpha_S . S_i^{\beta_S} \qquad (2)$$

Here α_J represents the relative weight of factor J in utility of the consumer, and β_J represents the steepness of relationship between different utility determining factors and overall utility. For example β_F, β_C, and β_S are typically between 0 and 1 (that is, there is a decreasing return on features, complementary software and on installed share), while sensitivity to price is only restricted to positive values. The inputs to utility function (F, P, C, and S) are normalized by market feature richness, maximum price, total complementary software, and total installed base, so that they all remain bounded and robust in the simulations that will follow.

To include the impact of "Openness Decision" (*O*) and "Pricing Decision" (*PD*) on price, we use another parameter, κ >0, to determine a maximum feasible price given a level of openness $(1-O^{\kappa})$, and the pricing decision (between 0 and 1) specifies the price to be charged between zero and this maximum feasible price:

$$Pi = (1-O^{\kappa}).PD_i \qquad (3)$$

Connecting the impact of software sales (*SS*) to installed share (*S*), we close the "R1: Network Effects" loop. Software sales accumulates in the installed base (*B*) which is drained with some time constant as consumers stop using the product, and installed share is simply the share of each player of the overall installed base in this market:

$$B_i = \int (SS_i - B_i / \tau_1).dt \qquad (4)$$

$$S_i = \frac{B_i}{\sum B_i} \qquad (5)$$

This feedback mechanism captures both the word of mouth effect and different network externalities that depend on the installed base of the software: the more the installed base, the higher the utility, and thus the market share and sales.

The introduction of "Complementary Software" (*CS*) depends both on the effect of the openness of the software (*EO*) and the installed share (*S*). The dependence on openness follow the observation that open platforms encourage complementary software developers to build their products on those platforms. On the other hand, installed share of products is typically a major motivation for complementary product developers to build products compatible with the bigger players in the market. Moreover, the complementary software has a useful life after which it looses its impact on the attractiveness of the focal software. Finally, Complementary Software Share is determined by the number of complementary software products for all the players. Specifically:

$$CS_i = \int (ic_{Max}.EO_i.S_i^{\varepsilon_S} - CS_i / \tau_2).dt \qquad (6)$$

$$EO_i = eo_{Min} + (1 - eo_{Min}).O_i^{\varepsilon_O} \qquad (7)$$

$$C_i = \frac{CS_i}{\sum CS_i} \qquad (8)$$

Parameters ε_O and ε_S determine the strength of the impact of openness and installed share on the introduction of complementary software. The parameter eo_{Min} (between 0 and 1) represents the rate of development of complementary software for completely closed platforms, as a fraction of complementary development rate for completely open platforms with otherwise identical characteristics.

The link from installed share to complementary software and back to sales introduces a second reinforcing loop: "R2: Complementary Software and Standardization". This loop captures the process of development of complementary products and the standardization of the software as a trusted platform, both of which follow the installed base with a delay, and thus are captured in a separate loop.

The installed share of the product also impacts the attractiveness of product for involvement of open source contributors (*OC*). The larger the current base, the more attractive the platform becomes for new open source developers as they see a larger prospect for impact and visibility of their work. For example Wikipedia is far more attractive for potential contributors than a small open source product with a few hundred installations. Besides the installed share, the current level of features under development determines a "Desired Joining Rate" (*DJ*) which limits the number of open source contributors joining the community. Finally, by definition the level of openness of a product has a significant impact on the desire of the potential open source contributors to join the community. The community members leave after some average residence time. Specifically:

$$OC_i = \int (Min(DJ_i, oj_{Max}.ES_i.O_i^{\upsilon_O}) - OC_i / \tau_3)dt \qquad (9)$$

$$ES_i = es_{Min} + (1 - es_{Min}).S_i^{\upsilon_S} \qquad (10)$$

The number of formal employees (*E*) is adjusted towards a desired employee level. The latter is determined based on the development resources, beyond open source community, needed to develop the features under development and the resources needed for service and maintenance of the current installed base. Finally, the profit stream (*R*) depend on the costs (employee costs ($c_E.E$) and fixed costs(c_{fix})) and revenue from both sales and services ($B_i.p_{Srv}$). We use the net present value (*NPV*) of this profit stream discounted to the beginning of product life cycle with discount rate *r* as the main performance function for the firm:

$$R_i = A.MS_i.P_i.p_{Max} + B_i.p_{Srv} - c_{fix} - c_E.E \qquad (11)$$

$$NPV_i(t) = \int_0^t e^{-r.s} R_i(s).ds \qquad (12)$$

Full model with complete variable listing is available in the supplementary material for replication and further analysis.

3-Analyzing pricing and openness policy for two competing firms

In this section we first provide a base run for the model to create a basic intuition about the dynamics involved. We then show the results of one firm optimizing its pricing and openness decisions assuming that the other firm behaves as in the base run. This analysis provides further insights into what policies can help one firm to succeed and the mechanisms responsible for effectiveness of those policies. We then layout the general process for finding the game theoretic equilibrium for this competition and share the results under the base case parameters of the model. Finally, we discuss the robustness of these results to different market and product characteristics.

3-1-Base run: Two firms with fixed policies

In the first experiment we simulate two identical firms with a fixed pricing policy (charge maximum feasible price, i.e. *PD*=1), and a mixed openness policy: *O*=0.5. Given the identical nature of the two firms, their performance over time is identical, therefore we only show the variables from one of the firms in Figure 2. The behavior of the system is partly driven by the exogenous Adoption Rate (Variable 3 in panel a) which follows a typical bell-curve pattern for the life cycle demand for the product. Installed base of software (Variable 2, panel a) follow the adoption, and complementary software (variable 2, panel a) lags further. Revenue (variable 1, panel b) is generated by a linear function of adoption (i.e. sales) and installed base. The costs (variable 2, panel b) on the other hand follow a significant initial rise to hire required employees (variable 1, panel 3) for the development of new initial features, and later goes down to steady state levels dictated by employees required for maintenance and service. Open source contributors (variable 2, panel 3) follow a similar pattern as employee do, but go to zero as service and maintenance are assumed to be done through contracts and by formal employees. The net present value (variable 3, panel b) goes down initially as profit is negative, but later starts to recover when sales and maintenance revenue dominate the costs. We focused on a 10 year period for analysis, until the profit tends to move towards zero at the end of the life cycle and the company would find the continuation of this line of business no more attractive. Given the identical nature of the two firms in this case, the impact of different feedback loops on their competition is not salient.

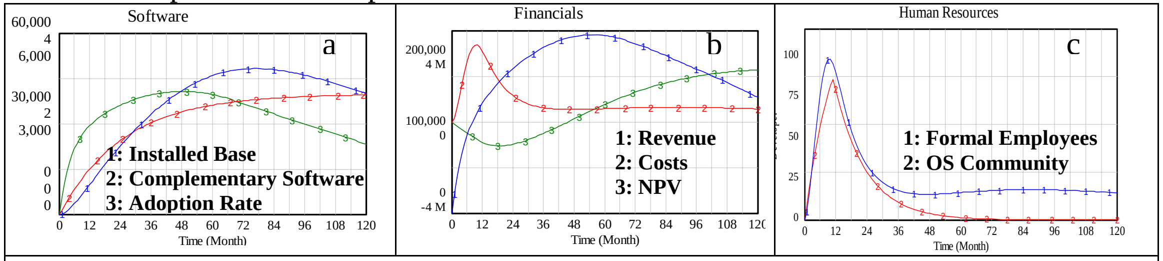


Figure 2-The base case behavior of the model with two identical competitors with no strategic behavior. a)Customers, complementary software, and adoption rate. B)Revenue, costs and net present value c)employees and open source contributors.

3-2- Optimizing a single firm's pricing and openness

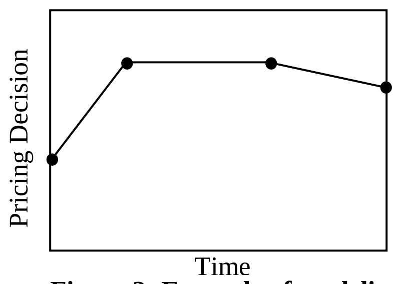


Figure 3- Example of modeling dynamic prices. Two parameters specify each point. Connecting these points provide the overall price at any point in time.

In the second experiment we allow one firm to optimize its pricing and openness decisions to maximize its final net present value (*NPV* at time 120), assuming that the other firm does not change its pricing or openness policies. In formulating the optimization problem openness is assumed to be constant throughout the simulation time. This is consistent with the nature of openness parameter which once decided, is hard to change due to legal and technical challenges. On the other hand, pricing policy could be dynamic. In extreme, the prices could change at any point in time. We simplify this problem by allowing the firm to select N points in time, including beginning and end times, with their respective prices. Price at any point in time is then determined by a linear weighing of the two points before and after that point in time. Figure 3 provides an example with N=4. In the reported experiments we use N=9, which we found to provide sufficient degree of flexibility for all the cases we analyzed. This setting leads to 17 parameters to be estimated for the firm: 1 for openness, 7 for the points in time (other than initial and end times) when pricing decision changes slope, and 9 for the pricing decision at the N points. We use Venism ™ optimization engine for solving this nonlinear optimization problem.

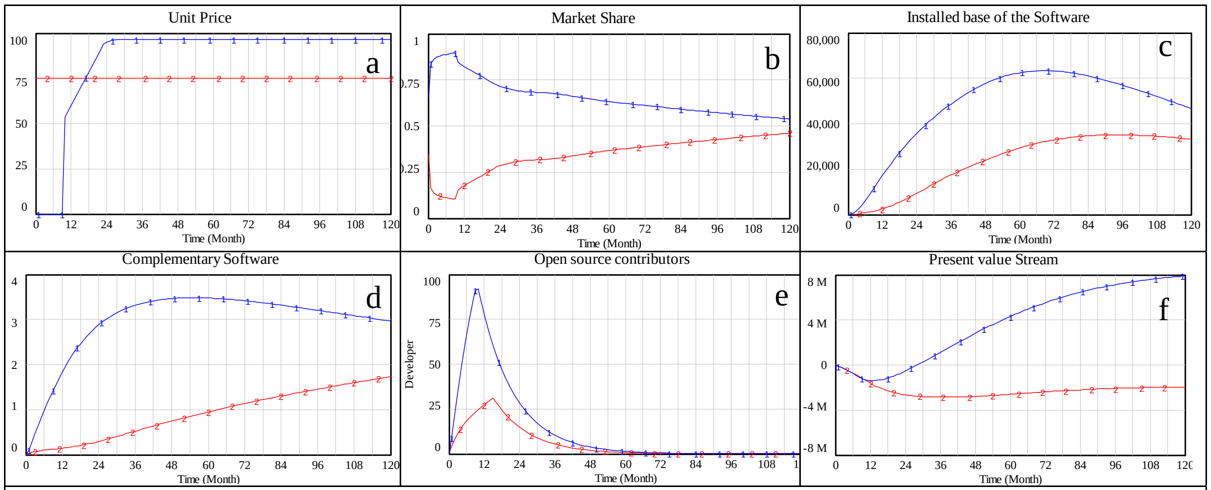


Figure 4-Performance of two firms when firm 1 is optimizing its price and openness

Figure 4 reports the results of optimizing firm 1's net present value, keeping the base parameters for firm 2. In this case optimum openness comes out at 0.36 (vs. 0.5 for firm 2) and optimum pricing is reported in panel a. The pricing decision (*PD*) starts from free distribution of the software for the first few months, to a relatively steep hike in the price so that by month 26 the company is charging the full price feasible (PD_1=1 for t>26). Given the lower openness of the software, firm 1 can charge more than its competitor, thus the higher final unit price. The initially strong price advantage for firm 1 leads to significant market share advantage early on for it (panel b). This helps firm 1 build a strong installed base (panel c) and thus attract many more complementary software developers than its competitor (panel d). It is interesting to note how a significant initial installed base also helps firm 1 receive many more open source contributors, leading to even further cost advantages despite the fact that firm 2 has a more open architecture (panel e). Another interesting feature of this experiment is the fact that by increasing the price firm 1 actually reduces its market share advantage over time in return for getting a more significant profit margin (panel b). Overall, firm 1 can earn a much higher return on its investment and reach a significant net present value where as firm 2 in this setting becomes profitable only after 3 years and can never reach a positive net present value due to significant losses early on (panel f).

The reinforcing loops of open source support, network effects, and complementary software significantly impact the shape of optimum pricing policy. In fact in the absence of these loops the optimal pricing and openness patterns for the firm will completely change. The optimum policy for firm 1 when the reinforcing loops R1, R2, and R3 are deactivated (by setting parameter β_C , β_S , and υ_S to 0) is to keep the software largely closed (O_1=0.06) and charge the highest feasible price (PD_1=1) . In fact under these conditions firm 1 *looses* market share to the second firm (because it is now offering a more expensive product), yet it improves its profitability by having much higher margins. Interestingly, firm 2 also benefits from this policy of the first firm as it gains market share and thus increases its revenue given its constant price. Overall, the reinforcing loops of network externality, complementary goods, and open source support increase the usefulness of open source policies and steep discounts early on.

3-3-Strategic competition between two firms

In the previous section we assumed firm 1 optimizes its price and openness while firm 2 does not make any adjustment to its base case policy. This is not a realistic assumption in practice because both firms probably have relatively similar opportunities for finding improved policies, and therefore they would both try to optimize their own performance, taking into consideration the actions of the other firm. In extreme, rational firms with full information about the structure of the competitor firm would find the pricing and openness policy that can not be improved upon if the competitor is also rational and looking for such policy. Should such policy exist, it constitutes a Nash equilibrium for the competition between the two firms. While the selection of Nash equilibrium by both firms may be behaviorally a strong assumption (given the computational, information, and coordination limitations of real organizations), finding this equilibrium is informative in telling us about the tendencies of the firm in adjusting its policy in such markets. Solving the equilibrium problem analytically is not feasible for the model at hand, therefore we use an iterative numerical method that was capable of solving the equilibrium problem for all the experiments in this paper.

The basic idea behind solving the strategic equilibrium problem is simple. Consider the experiments in the section 3-2, and assume that firm 2 predicts firm 1's optimal policy (that we found above) and using that as what firm 1 will do, optimizes its own pricing and openness policy. Next, firm 1 takes this policy of the second firm as input, and optimizes its own policy for a second round. This process can continue for as many rounds as needed until the two firms converge to the same policy[1]. If they do

[1] Convergence to the equilibrium (same policy) is not guaranteed, even if such equilibrium exists. Convergence may depend on starting points of the search process. Moreover, this process does not provide

converge to such policy, that policy is by definition a Nash equilibrium for the competition between the two firms: neither firm can improve its performance by deviating from this policy. In our computational experiments under different parameter settings the firms converged to the same policy after at most five iterations of the above process (a total of 10 optimizations).

Figure 5-a reports the successive changes in the optimum pricing policy for firm 1 (optimization rounds 1, 3, 5, 7, and 9), that is, item 1 represents optimum policy for firm 1 if firm 2 is following base case policy, item 2 representing optimum for 1 if firm 2 is following firm 1's policy in item 1, and so on. As the competition between the two firms get closer to a fully rational one, both firms tend to increase their openness (firm 1 going from O_1=0.36 to 0.53, 0.64, 0.68, and 0.69 in the five successive optimizations) and provide the software for free longer, in the hope of increasing their early market share advantage over their competitor. Higher discounts and openness however are copied by the other firm who is also seeking to improve its profits, leading both firms to receive a lower profit stream (panel b) in a prisoners' dilemma type of game. In fact with the current parameter settings net present value for neither firm becomes positive in the final equilibrium (item 5 in both graphs).

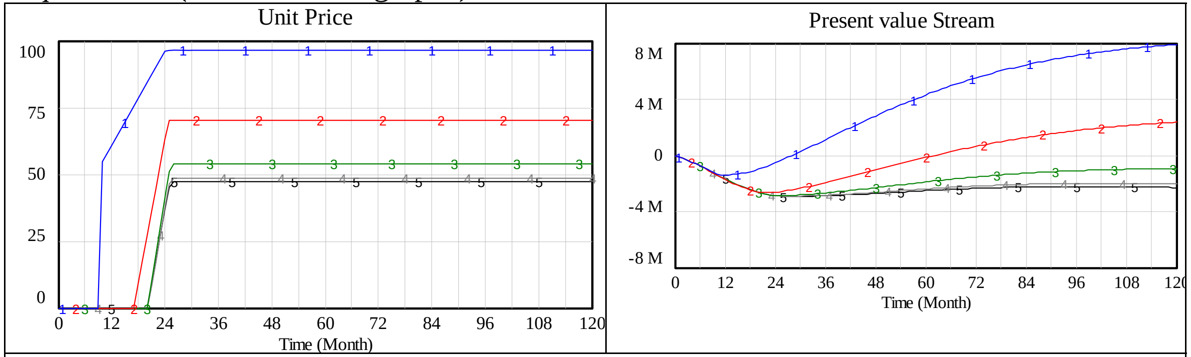
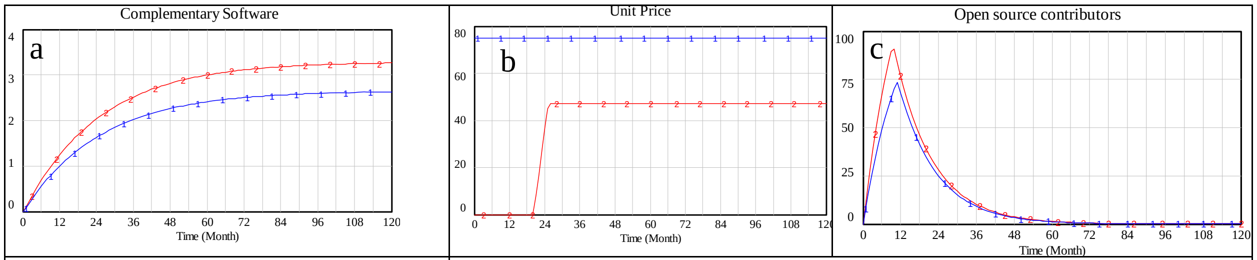


Figure 5- Successive iterations of optimization to find strategic equlibrium for both firms. Item 5 represents the optimal policy under strategic competition which is identical for both firms. a) The unit price for firm 1 over time b) The net present value for firm 1.

A comparison between the base case (Figure 2, where no competitive action was modeled) and the final strategic equilibrium is instructive (Figure 6). In both cases the market share is equally distributed between the two identical firms, therefore the installed base for both firms is identical to the base case (Figure 1-a). However, increased openness of the software in the competitive equilibrium leads to increased complementary software in this case (Figure 6-a), and reduced prices for both firms (Figure 6-b). These factors increase the overall utility of the customers benefiting from these products in this market. Moreover, open source developers become a more important part of the development process given their higher share in the product development process (panel c). On the other hand both firms are financially worse off in the competition (panel d). Overall, moving towards the competitive equilibrium tends to benefit the users of the software at the expense of the software producing firms, leads to more open platforms and lower prices or free products.

any direct way to assess the uniqueness of an equilibrium point. In all the experiments reported here however we did converge to a single policy regardless of the optimization starting point in the policy space.



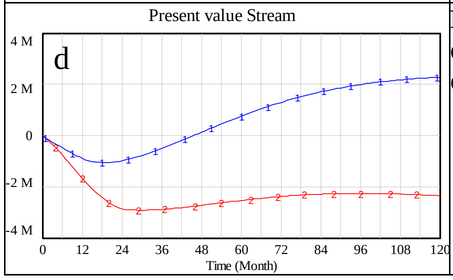


Figure 6- Comparison of base case (item 1) and competitive equilibrium (item 2) for complementary software (a), unit price (b), open source contributors (c), and net present value over time (d).

3-4- Robustness of results

The preceding analysis used a set of base case parameters to define the strength of different feedback processes and customer utility function. In this section we analyze how sensitive the competitive equilibrium results are to some key parametric assumptions. First we vary seven of the major model parameters around their base levels to assess the impact of different loops and effects individually. We then combine a few of these effects to see how completely different market conditions can change the optimal pricing and openness decisions. Table 1 reports the parameter settings used for this analysis.

Table 1- The parameter settings used in the sensitivity analysis. Full paramters and equations are available in the attached model.

Parameter	Base Value	Lower	Higher	Equation
Sensitivity of Utility to Price (β_P)	1	0.5	2	2
Sensitivity of Utility to Installed Share (β_S)	0.5	0.1	0.8	2
Sensitivity of Utility to Complementary Software Share (β_C)	0.5	0.2	1	2
Sensitivity of Complementary Software to Openness (ε_O)	1	0.5	2	7
Sensitivity of Complementary Software to Installed Share (ε_S)	1.5	0.5	3	6
Sensitivity of Open Source Contributors to Installed Share (υ_S)	1.5	0.5	3	10
Sensitivity of Price to Openness (κ)	1	0.5	2	3

Figure 7 reports the optimum pricing policy and openness for each sensitivity analysis as compared to the base case. Item 3 in each graph represents the base optimum policy, item 2 represents the results with the higher value of the parameter, and item 1 represents results with the lower value. In general, all equilibrium policies continue to have the S-shaped feature of starting from fully free software and moving towards charging the maximum price feasible at some point in time. The sensitivity of results to different parameters, however, differs. For example, how sensitive the open source contributors are to the installed share of the software has little impact on the competitive equilibrium (panel f). In contrast, the sensitivity of the customer utility to complementary software makes a significant difference in the magnitude of openness in equilibrium (panel c): low sensitivity leads to relatively closed software (O=0.45) compared to when customers really care about complementary software.

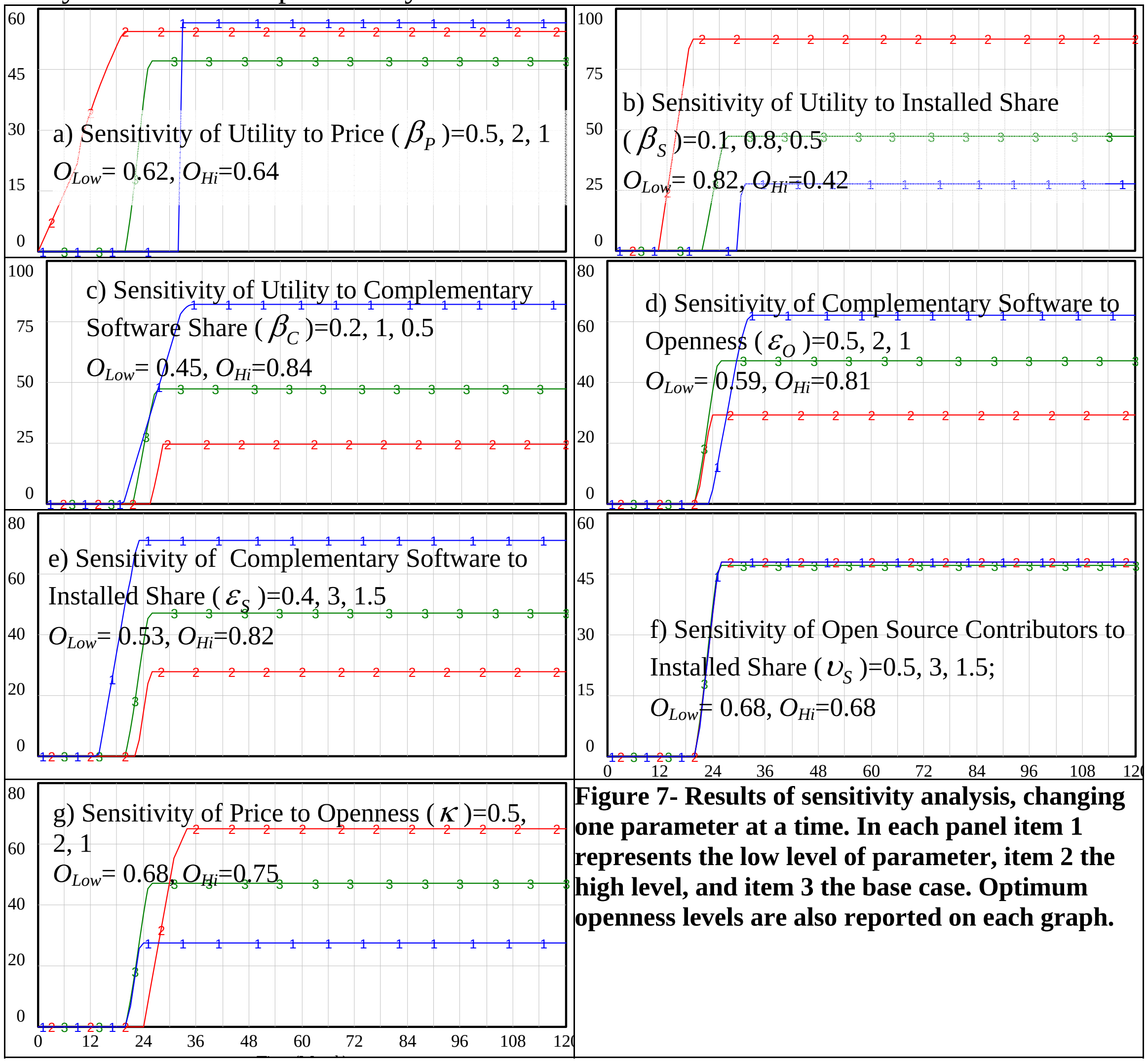


Figure 7- Results of sensitivity analysis, changing one parameter at a time. In each panel item 1 represents the low level of parameter, item 2 the high level, and item 3 the base case. Optimum openness levels are also reported on each graph.

Sensitivity of utility to installed share (panel b) also shows a significant impact: the higher this sensitivity, the more openness is induced in the equilibrium. Therefore one can expect that in markets with significant word of mouth effects and network externalities, the balance tilts towards companies with more open platforms and lower prices. Weak network effects promote more proprietary platforms with higher prices. The strength of reaction of complementary software developers to openness is also influential: the stronger this reaction, the more open the equilibrium platform (panel d), resulting in lower prices. Such sensitivity is likely to be higher when complementary software should be directly integrated with the focal software, such as integration of social network platforms with their applications. In contrast, modular complementary software that does not need access to internal workings of a software is less sensitive to its level of openness (e.g. most operating systems). Complementary software is also influential through its sensitivity to installed base: when complementary software producers want to only work with the market leader, the overall market dynamics favors open source and free products (panel e). Again, this would typically be the case when the focal software integrates with the complementary products and therefore a lot of development work for complementary product is unique to the platform in question.

These effects suggest that the stronger the reinforcing loops of network externality and complementary products, the more the market tips towards the open source business model where price is low (or product is free), most of the work is done by the open source community, and the major source of revenue is the service and maintenance fees. The third reinforcing loop, open source support, proves less consequential however. This loop acts through the design of new features. Yet in the absence of open source developers the company will hire enough paid employees to keep up with the market. As a result the real effect of openness in gaining market share does not come from the use of open source community, rather, through the reduced price of the product when it is open source, and through the impact of complementary products.

Two other experiments need further explanation. First, as panel a shows, the main difference between higher and lower sensitivities of utility to price is in the timing of price change. With high parameter values (β_P =2) the equilibrium price is increased earlier (in fact from the beginning of the simulation) compared to the low β_P case in which equilibrium price increases more abruptly and later than base case. To understand this effect we need to note that in equation 2 β_P operates on a price metric that is normalize against its maximum value and thus is always smaller than one. With higher than one β_P and smaller than one price index (P_i), the impact of utility on price will remain very small compared to other utility factors. For example P_i of 0.2 vs. 0.3 leads to effects of 0.04 vs. 0.09 when β_P is 2, both very small effects. As a result the model allows for faster and earlier increase in the price. Yet the overall price is largely bound to openness level which remains fairly high in light of its impact on complementary software development. Therefore intuitively, price sensitive customers also further push the equilibrium of competition towards free software. Finally, the sensitivity of price to openness reflects how much price should be reduced if openness is increased. A low value for κ leads to a strong reaction (see equation 3 and note that O is smaller than 1) while a high value leads to less impact. As a result higher prices are feasible with higher openness levels when κ =2, leading to an equilibrium in which both price and openness increase (panel g, item 2).

These experiments, therefore, suggest that removing the reinforcing loops of network externality and complementary software (R1 and R2 in Figure 1) could change the optimum firm policy in this market considerably. To test this hypothesis we conduct an experiment in which both these feedback loops are cut, that is, $\beta_C = \beta_S$ =0, while all other parameters remain the same as those in the base case. The results are reported in Figure 8. As anticipated, the shift in the optimum policy is very significant: the optimum openness shifts to zero, while full price is charged from the beginning to the end (PD=1) for both firms. The firms are no more under significant pressure to sacrifice their profitability or open up their source code to gain an initial lead in the market share, and as a result they end up with significant improved profitability (panel b, compare to Figure 2). This comes despite the fact that the firm no more benefits from open source contributors (panel c). Moreover, with a fully closed architecture in a completely proprietary model fewer complementary software products are written for it (panel a).

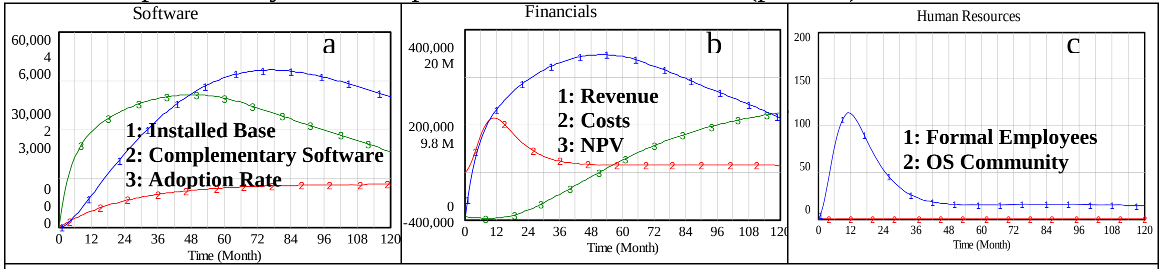


Figure 8-The competitive equilibrium in the absence of complementary product and network externality reinforcing loops a)Customers, complementary software, and adoption rate. B)Revenue, costs and net present value c)employees and open source contributors.

4-Discussion

In this paper we analyzed pricing and openness decisions for software firms exposed to different reinforcing loops in the software market. A few general themes emerge. First, the network and complementary product effects significantly influence the desired firm policy. When strong reinforcing loops are present, the firms need to do all it takes to build an initial installed base and attract complementary software producers. It can do so by both giving the product for free, and opening up its platform to benefit from open source contributors and encourage complementary developers. The pricing will later be shifted towards charging for the software, yet the charged amount is much less given the openness of the product that does not allow for high prices. These themes are most salient when we consider the strategic competition among two firms, where the Nash equilibrium of their competition dictates significant openness and discounts. When network and complementary effects are absent, optimum policy could favor the completely proprietary model with maximum chargeable price.

Secondly, deciding on the right level of openness is tightly coupled with the pricing decision and the reinforcing loops discussed. The increasing popularity of open source software has motivated many commercial software producers to look into profitable opportunities in alternative business models that benefit from openness. Our results suggest that those opportunities are limited: openness, as a business move, is beneficial where strong reinforcing loops of network and complementary product exist, yet the very same market structures require significant discounts that hurt profitability significantly. Head to head competition therefore significantly limits potential returns in these markets.

On the other hand, if a firm benefits from early mover advantage, which allows it to grow significantly before competitors consider the market, openness could prove a profitable business move. This is because openness leads to strong barriers to entry for later competitors and thus allows the firm to rip the benefits of a large installed base (e.g. through service and maintenance) without coping with all the competitive pressures. Gaining such early mover advantage however is not easy. In a market replete with smart entrepreneurs, where information travels very fast, players have access to similar technologies, and capital costs are limited, good ideas rarely remain secrete of one player. Therefore despite the huge buzz around success stories of get-big-fast strategy in emerging online market, from Amazon to YouTube and Facebook, the fact is that only a handful of firms have successfully traversed this path and for each successful case there are hundreds of failed businesses.

The research also provides some insights about the structure of software markets with strong reinforcing loops. Early in the market life cycle many things remain unknown: what is the market size, how many competitors will join, and what prices could be charged to make the venture profitable. Bright ideas and promises of fast growth can lure investors and entrepreneurs into exploring such market opportunities without full appreciation of the odds involved in light of the competitive pressures above. Start up firms start with deep discounts and free products to gain the early advantage in the market, hoping that soon they will be able to switch to a profitable model by charging higher prices for their product, or for the complementary services. Yet as the competitors join the market with their own discounts, these plans have to change and profitability should be delayed in hope of driving the competition out. The fierce competition often continues until seed money runs out and investors pull out. Sometimes in this time frame a firm reaches large enough a market share and size that it can sustain its business and become one of the few success stories. More often however, all the players find the market niche to be too costly to make it worth the effort. Such markets may completely be abandoned as a result. On the other hand, the price sensitivity of the consumers in this market increase as they get used to products that are free when they are the most novel. Such increasing price sensitivity further reinforces the dynamics above. It is not clear how such market structures would impact long term consumer utility: on the one hand discounts and large complementary networks of products benefit consumers significantly, on the other hand the dynamics discussed could keep more mature players from ever addressing some needs and market opportunities. So far the consumers have largely benefited from these trends, it should be seen whether the negative effects will catch up, e.g. through a crash in different Web 2.0 market that has been a hot market with significant reinforcing loops.

The framework and the model developed in this paper can also be used for analyzing the options available to a specific firm in a specific market. Such analysis requires empirical estimation of multiple parameters of the model based on the market and firm characteristics. In also requires insights into the decision making process and firm characteristics of competitors. This data may be hard to obtain, but the resulting analysis can move a long way towards better market entry decisions and pricing and openness policies for incumbent firms.

This research is limited on multiple fronts. First, the analysis focused on a constant market life cycle that is not impacted by the different factors influencing the utility of consumers. In practice the market also grows (or shrinks) depending on the features, quality, price, complementary products, and other characteristics of the competition. Inclusion of the feedback to market size can potentially change the structure of optimum policies for firms involved. Several other relevant dynamics, such as learning effects where also excluded in this analysis.

Furthermore, we focused on two identical competitors starting at the same time in this market. Entry timing, competitor heterogeneity, and multiplicity of competitors can change some of the dynamics discussed. For example late entrants, facing strong competition, may continue to provide discounts on their product, pushing the other firms to retain low prices longer. Despite these limitations, we hope our analysis provides some insights into the complex inter-relationship of openness, pricing, and reinforcing feedback loops in software business.

References:

Argote, L. and D. Epple (1990). "Learning-Curves in Manufacturing." Science **247**(4945): 920-924.

Arthur, W. B . (1989). "Competing Technologies, Increasing Returns, and Lock In by Historical Events." Economic Journal **99**(1): 116-131.

Cabral, L. M. B. and M. H. Riordan (1994). "The Learning-Curve, Market Dominance, and Predatory Pricing." Econometrica **62**(5): 1115-1140.

Dellarocas, C. (2003). "The digitization of word of mouth: Promise and challenges of online feedback mechanisms." Management Science **49**(10): 1407-1424.

Dodson, J. A. and E. Muller (1978). "Models of New Product Diffusion through Advertising and Word-of-Mouth." Management Science **24**(15): 1568-1578.

Facebook. (2009). "Facebook Statistics." Retrieved March 11, 2009, from http://www.facebook.com/press/info.php?statistics.

Frank, R. H. and P. J. Cook (1995). The winner-take-all society : how more and more Americans compete for ever fewer and bigger prizes, encouraging economic waste, income inequality, and an impoverished cultural life. New York, Free Press.

Hawkins, R. E. (2004). "The economics of open source software for a competitive firm." Netnomics **6**: 103-117.

Katz, M. L. and C. Shapiro (1992). "Product Introduction with Network Externalities." Journal of Industrial Economics **40**(1): 55-83.

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

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

Nilendu, P. and T. R. Madanmohan (2002). Competing on Open Source: Strategies and Practices, Indian Institute of Management-Bangalore.

Oliva, R., J. D. Sterman and M. Giese (2003). "Limits to growth in the new economy: exploring the 'get big fast' strategy in e-commerce." System Dynamics Review **19**(2): 83-117.

Panzar, J. C. and R. D. Willig (1981). "Economies of Scope." American Economic Review **71**(2): 268-272.

Ratnarajah, T. (2008). Modeling the dynamics of software competition to find appropriate trade-off between openness and price. Industrial and Systems Engineering. Blacksburg, Virginia Tech. **MSc**.

Spence, A. M. (1981). "The Learning-Curve and Competition." Bell Journal of Economics **12**(1): 49-70.

Sterman, J. (2000). Business Dynamics: systems thinking and modeling for a complex world. Irwin, McGraw-Hill.

Wu, M. W. and Y. D. Lin (2001). "Open source software development: An overview." Computer **34**(6): 33-+.