Competition and innovation in a technology setting software duopoly

Jürgen Bitzer* Philipp J.H. Schröder†

March 2004

Abstract

Recently the software industry has experienced fundamental changes in market structure through the entry of open source competitors, e.g. Linux's entry into the operating systems market. In a simple model we examine the effects of such a change in market structure from monopoly to duopoly under the assumption that software producers compete in technology rather than price or quantities. The model includes the presence of technological progress and menu costs of adjusting existing software, i.e. innovation. It is found that: (i) moving from monopoly to duopoly does increase the technology level set by firms in the software industry; (ii) a duopoly adjusts more readily to global technological progress than a monopolist. Furthermore, results are presented comparing open source versus for-profit firms in terms of technology levels and innovation.

Keywords: open source software, strategic interaction, duopoly, menu costs.
JEL classification: H41, L86, L31

1 Introduction

The emergence of open source software (OSS) as competing products in formerly monopolistic software markets has led to a discussion on the consequences of competition caused by OSS on innovation and the rate of progress

*Free University Berlin, German

†Aarhus School of Business, Denmark and DIW - Berlin. Corresponding author: Philipp Schröder, Aarhus School of Business, Fuglesangs Alle 4, 8210 Aarhus v, Denmark, e-mail: psc@asb.dk. The Authors thank Roger Sherman and Zhentang Zhang for valuable comments. The usual disclaimer applies.

in software technology development.[1] Consider the comments of former *de facto* monopolists.

> "Open source is an intellectual-property destroyer, I can't imagine something that could be worse than this for the software business and the intellectual-property business. (...). I'm an American, I believe in the American Way. (...). I worry if the government encourages open source, and I don't think we've done enough education of policy makers to understand the threat."[2]
>
> (Jim Allchin, Microsoft Group Vice President Platforms, cited after Bloomberg, 2001)

Or the following remarks concerning the efficiency of the OSS development process:

> "The OSS development model leads to a strong possibility of unhealthy 'forking' of a code base, resulting in the development of multiple incompatible versions of programs, weakened interoperability, product instability ..."
>
> (Craig Mundie, Microsoft Senior Vice President, talk given at New York University Stern School of Business, cited after Microsoft, 2001).

Looking through the dust swirled up by a troubled monopolist there remain two economic arguments which claim that the emergence of OSS competition in the software market could lead to reduced economic efficiency and welfare: (i) OSS, as a public good, allegedly destroys intellectual property and thus reduces incentives to innovate, with negative consequences for welfare. (ii) The loose coordination of the innovation process in the development of OSS leads to redundant developments and a less efficient method of software production, which lowers welfare. While the first argument is derived from competition between a commercial software enterprise and OSS developers, the second is derived from competition among OSS products. Both suggest sluggish technological progress.

[1] For an introduction to the history and success of OSS see for example Stallmann (1999), Rosenberg (2000), Feller and Fitzgerald (2002) or Hars and Ou (2002).

[2] In a later amendment of Mr. Allchin's comments, Microsoft stated that it only had a problem with the "Marxist-Leninist tool, the GNU GLP license" and not with the open source movement as such (Chaffin, 2001). There are even harsher statements, e.g. Steve Ballmer, Microsoft CEO, is quoted by the Chicago Sun-Times (2001) as saying that Linux is " (...) a cancer that attaches itself in an intellectual property sense to everything it touches."

Data on technological progress in software technologies does not lend support to either of these arguments. Rather it seems that the rate of commercial software and OSS innovation has increased with emerging competition. The release history[3] of Microsoft's Windows (see Table 1) shows that the intervals between major releases have decreased continuously since the beginning of the 1990s.

Table 1: Release history of MS Windows[4]

Release date	**Version**
1985	Windows 1.0
1987	Windows 2.0
1990	Windows 3.0
1995	Windows 95
1998	Windows 98
2000	Windows ME
2001	Windows XP

Source: Information published at: [www.microsoft.com].

The same is true for competition among OSS projects, as documented in Tables A1 and A2 for GNOME and KDE. The case of the two OSS competitors GNOME and KDE shows that competition[5] did not reduce the rate of innovation. On the contrary, both branches appear to have gained momentum once competition kicked in (see Figure 1).

The majority of the OSS economics literature addresses the "incentives to innovate/participate" issue, i.e. the question of the motivation of OSS programmers and the capacity of OSS software (a privately provided public good) to offer viable alternatives to commercial software products. On this, see e.g. Lerner and Tirole (2002); Johnson (2002); Raymond (2000a b); Hars and Ou (2002); Bitzer and Schröder (2002). Furthermore, a few papers address the competition between OSS and commercial software, e.g.

[3]The release history can be regarded as a suitable indicator of technological progress in software technology. This is because the release of a new version creates costs, not only for the software firm or OSS developer but also for the customers implementing it. A customer will only be willing to invest in the new software version if the benefits thereof are greater than the implementation costs. Thus, the resulting release costs will only be justified if there is at least a certain technological improvement in newly released software. Furthermore, it can be expected that the higher the release costs, the higher the technological improvement necessary to offset investment.

[4]Further minor releases have been Windows 3.1 and Windows for Workgroups 3.11 in 1993 and Windows 98 SE in 1999.

[5]Another common case of competition within OSS projects is caused by forking, i.e. the spliting of a project into two competing branches.

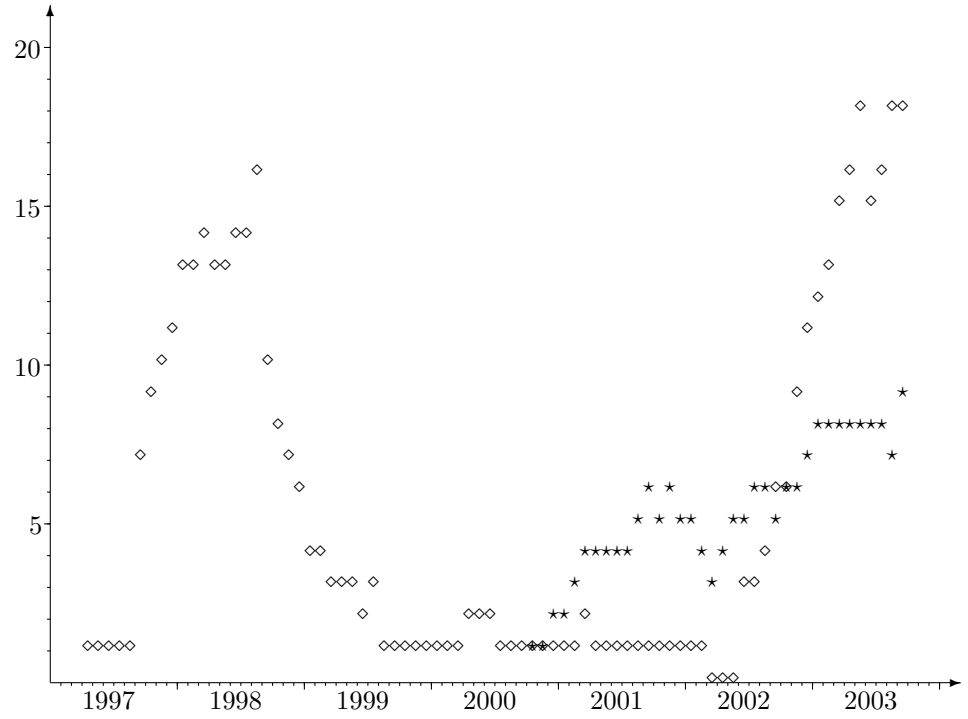

Figure 1: Number of Version Releases in the Past 12 Months, Gnome:◇ / KDE:⋆ Source: Appendix Tables A1 and A2

Casadesus and Ghemawat (2003); Bitzer (2001); Dalle and Jullien (2002).[6] Despite impressive progress in research on the OSS phenomenon (see also the special issue of Research Policy on OSS, forthcoming in 2003) to the best of our knowledge no study thus far has addressed the "sluggish technological progress" argument mentioned above.[7] The literature on OSS has not yet dealt with the influence of an increase in competition through the emergence of OSS upon the rate and level of technological advances in a software monopoly/duopoly.

This paper addresses the impact of a change in competitive structure on the technology development decision of for-profit firms and OSS developers in the software market. We examine this issue both in terms of setting the technology level of a software product, and in terms of the willingness to react to external technological advances (shocks).[8] To capture both OSS and

[6]The papers of Casadesus/Ghemawat (2003) and Bitzer (2001) focus on the question of which software will ultimately "win" the competition and eventually displace the competitor. Dalle/Jullien (2002) address the specific role of standards in the competition between commercial software companies and OSS developers.

[7]By the same token, no paper has yet examined the consequences of competition (e.g. caused by forking or branching) on the innovation activity within OSS projects.

[8]The paper does not deal with possible constraints to the supply of innovation, e.g. the emergence of new firms in the software industry or the appearance of 'forks' in OSS

for-profit software producers within the same formal framework, we abstract from profit, altruistic or job-signaling motives and use a general objective function based on the dissemination of a software product and its technological level. Thus, producers are assumed, be they OSS or for-profit, to value the extent to which their product is distributed/bought. The demand for a certain piece of software, i.e. its dissemination, in turn, is assumed to be neither a matter of price nor of available quantity, but rather to depend on the technological content (level) that the software offers to users. Thus, *technology* is the strategic variable of software competition, and, while a high level of technology ensures wider dissemination, it is also associated with a higher cost to the developer. Furthermore, software technology is subject to exogenous technology shocks. The 'real' technological level of any piece of software quickly deteriorates as externally determined technological possibilities (processor capacity, application demands, etc.) and consumer demands grow. Therefore the technology embedded in a developed piece of software must be adjusted in accordance with external (global) technological progress. In turn, these adjustments (e.g. releasing an updated software version) entail particulare costs (e.g. development and release costs). Using the framework of Rotemberg and Saloner (1987), these adjustment costs are formally introduced as a menu cost.

The paper derives the following results from the formal model: First, the transition from a monopoly to a duopoly increases the technological level chosen by the enterprises. Second, the transition from a monopoly to a duopoly increases the willingness to adjust to global technological shocks (rate of innovation). These findings apply both to pure commercial software and pure OSS markets as well as to mixed markets (e.g. entry of an OSS firm into the market of a for-profit monopolist). Third, assuming that development and innovation costs of OSS firms are lower than those of for-profit firms, pure OSS duopolies will display more advanced technologies and a higher rate of innovation. Yet, for a sufficiently strong profit/payoff motive, the inverse conclusion holds. These findings confirm the above mentioned observations on the software industry.

The remainder of the paper is as follows. The following section introduces the formal model and discusses its implications. Section 3 concludes the paper.

---

projects reducing the number of programmers working on each individual product/project, thus reducing the rate of innovation via a supply-side effect. This type of argument has also not been advanced either by Microsoft or others. Hence our focus is on the ways that strategic interactions among technology setting software firms will affect innovation, and not on the effect of splitting the possibly limited resources in human/programming capital among competing products.

2 The Model

2.1 Software competition

Competition between a profit oriented commercial software firm and an OSS developer community follows different rules than the "standard" competition model. First, the incentives of the actors differ. Second, the strategic variables are neither price nor quantity, but rather technology. Third, the behavior of the market participants is strongly influenced by an exogenous technological factor.

First, the incentives of OSS developers have recieved a great deal of attention. It is widely acknowledged that, while commercial firms maximize profit, OSS developers are interested in enhancing their reputation and/or signalling value (e.g. Raymond, 2000a, 2000b; Torvalds and Diamond, 2001; Lerner and Tirole, 2002). Hence, even though a profit motive can be ruled out for OSS developers, they nevertheless maximise these other payoffs. Although the incentives of the two types of software producers are different, both incentives are positively correlated with the dissemination of their respective software product. While commercial firms are interested in increasing their profits by benefitting from decreasing average costs with increasing dissemination of their software, OSS developers benefit from the dissemination of their OSS in terms of enhanced reputation and signalling value (Lerner and Tirole, 2002). Thus, commercial firms and OSS developers can still be assumed to maximise their respective payoffs, which depend in turn on the dissemination of their software. To capture both types of software producers, we use a general objective function that can represent both commercial software producers and OSS developers.

Second, the strategic variable in a software market is neither price nor quantity. Software is an intangible good that can be duplicated at virtually no cost, and in addition, at least one agent (the OSS developer) distributes his product at zero price.[9] Since there can be no talk of either quantity *or* price competition, our paper starts out by formulating a (admittedly unconventional) model in which competition is not based on price or quantities, but instead software providers compete in technological advancement of their products. A high level of technology ensures widespread dissemination of the software, which is good for the producer, but it also raises the costs of development and maintenance (bug fixing etc.). We apply a broad concept

[9]In fact also the price of most commercial software is often inessential from a consumer's point of view. The majority of software is sold as pre-installed, thus its ability, reliability, compatibility, etc – in short its technological content – drives the consumer's decision, while the price is a matter between the soft- and hardware producer.

of technology including all properties which influence the user's decision to employ a certain software package. Depending on the type of software, the technology therefore includes characteristics like supported hardware, ease of use/installation, interconnectivity capabilities, range of features, state-of-the-art functions, performance, quality, reliability, and so on. Thus, the technology of a software includes the entire bundle of its technological characteristics.

These considerations lead us directly to the third difference between our model and a "standard" competition setting. The "value" of a piece of software to a user depends strongly on how up-to-date its general functionality is or, to put it differently, its "real" technological level. The real technological level depends on how far each technological characteristic of the software is behind the state of the art: the "technological frontier" of that particular aspect of the software. Thus, the technological level of any piece of software is defined in relation to the global technological level, which consists of all the most advanced developments in each aspect of that software at the current point in time.[10] On the other hand, the global technological level itself is constantly changing. It is set by the developments in the globally available technology which influences demand for or development of software. The global technological level is driven by developments in hardware technology, new applications, development of new features, consumer demands and so on.

2.2 A simple framework

Consider a heterogeneous goods software duopoly. The two firms a and b service the imperfectly separated market segments A and B respectively. Instead of competing on price or quantity, the two firms compete in the technology of their respective software products. Hence the strategic variable is the technological level of the product, or rather, innovation and development. At time t the technological level of firm i's software is denoted by τ_t^i , $i = a, b$. Further, the global technology level, T_t, represents hardware advances, new applications or changes in consumer demand.

The demand for software – or rather, the dissemination of the two software products – is assumed to be symmetric and to depend on the technological advance/ability of the software, and on the interdependence between the two market segments, i.e. the two products are imperfect substitutes.

[10] As no single software product is at the technological frontier in terms of every one of its technological dimensions, the technological level of any specific piece of software must always be lower than the global technological level.

Dissemination, s_t^i, of the two products a and b at time t is represented by $s_t^i = max\{0, q_t^i\}$, $i = a, b$, where

$$q_t^a = \left(\frac{\alpha}{2} + \beta\right) \frac{\tau_t^a}{T_t} - \frac{\beta \tau_t^b}{T_t} \tag{1}$$

$$q_t^b = \left(\frac{\alpha}{2} + \beta\right) \frac{\tau_t^b}{T_t} - \frac{\beta \tau_t^a}{T_t} \tag{2}$$

Parameter $\beta > 0$ is a measure of the substitutability of the two products and α is a positive constant. The expressions $\frac{\tau_t^i}{T_t}$ represent the "real technology level" of firm i. Namely, how advanced the technological capability of the software product is in relation to the hardware ability, consumer demands etc. Thus (1) and (2) state that the extent to which a certain software producer's product is distributed/sold depends positively on its own real technology level and negatively on the competitor's real technology level. Equations (1) and (2) also include the effect of external technological development in T_t which devalues the real technology level and dissemination of both software products.

Even though the model avoids the notion of price or quantity competition, the principles of maximisation are still applicable. Assume that firms derive some payoff from each distributed/sold unit of software. In particular there is a gain ρ, which could represent the reputational or signaling value for an OSS producer or more conventionally the per unit monetary reward for a proprietary software firm. There is also a cost C which is assumed to depend proportionally on the technology level of the product. Thus, a high real technology level is associated with high development and maintenance costs, i.e. larger support or hotline costs, costs of bug-fixing or indeed distribution and production costs. The latter in particular can be seen as driven by the fact that programmers working on a more advanced software product are more expensive than those working on an inferior software project. Postulating $C = c\frac{\tau_t^i}{T_t}$ the gain function for firm i can be stated as:

$$g_t^i = q_t^i \left(\rho - c\frac{\tau_t^i}{T_t}\right) \ , \ i = a, b\,. \tag{3}$$

How does this situation of software duopoly compare to that of a software monopolist? Assume that a monopolist is servicing both market segments A and B with the respective software products a and b. The gain-functions for the monopolist are identical to those formulated in (3). Yet the monopolist is aware of the interaction of the two markets, represented by β in (1) and (2), and takes this fact into account. In particular, due to symmetry, a

monopolist, M, realises that $\tau_t^{Ma} = \tau_t^{Mb}$. Rewriting the monopolists gain function for market i after setting in (1) and (2) respectively, gives

$$g_t^{Mi} = \frac{\alpha}{2}\frac{\tau_t^{Mi}}{T_t}\left(\rho - c\frac{\tau_t^{Mi}}{T_t}\right) \ , \ i = a, b \, . \tag{4}$$

Since both markets behave identically, in subsequent analysis it suffices to consider only one of the market segments.

2.3 Setting technology levels

In the case of a software duopoly, where firms behave noncooperatively and simultaneously have to choose their respective technology levels to maximise (3), the first order condition for firm a after substituting in q_t^a from (1) becomes

$$\tau_t^a = \frac{\rho T_t}{2c} + \frac{\beta \tau_t^b}{\alpha + 2\beta}, \tag{5}$$

and similarly for firm b.

Given that both firms expect all future technology levels T_{t+j} , $(j = 1, 2, \dots)$ to be equal to T_t, then the Nash equilibrium technology levels at time t are

$$\tau_t^a = \tau_t^b = \frac{\rho T_t(\alpha + 2\beta)}{2c(\alpha + \beta)}. \tag{6}$$

Lemma 1. *The technology level set by a software duopoly increases for a higher payoff, ρ, a falling cost, c, and a higher degree of substitutability, β.*

Thus, if we assume that an OSS duopoly has lower costs c compared to a for-profit software duopoly, then Lemma 1 states that an OSS software duopoly will produce a higher technological level. Similarly a duopoly with a higher payoff ρ, which may be the case for for-profit firms will settle for a higher technology level. Finally, once the two software products become more homogeneous (higher β) the strategic interaction in the software duopoly triggers firms to set a higher technology level.

Compare this to the asymmetric case. In a software duopoly with heterogeneous costs, c_a and c_b, firm a's Nash equilibrium technology level would become $\tau_t^a = \frac{\rho T_t(\alpha+2\beta)(c_a\beta+c_b(\alpha+2\beta))}{2c_ac_b(\alpha 2+4\alpha\beta+3\beta 2)}$, such that $\frac{\partial \tau_t^a}{\partial c_a} < 0$ and $\frac{\partial \tau_t^a}{\partial c_b} < 0$. A low cost competitor b increases the technology level set by firm a. Thus, we obtain:

Lemma 2. *A reduction in the cost of one firm in a software duopoly increases the individual technology levels set by both firms.*

In terms of Microsoft versus OSS, Lemma 2 implies that the entry of a low cost competitor pushes up the technology level of the for-profit firm beyond the level that would have resulted from a for-profit entry with higher costs, c_b. We do not pursue this point further and stick with our assumption of $c_a = c_b = c$ for the reminder of the paper, such that the impact from OSS on the for-profit competitor stems solely from the strategic interaction of the two firms rather than from the possible cost advantages that an OSS competitor may have.

Compare the above findings to a software monopolist maximising (4), and expecting all future technology level $T_{t+j}\,,\ j = (1, 2, \dots)$ to be equal to T_t. Such monopolist has first order conditions for market segment A and B that define the optimal technology level as

$$\tau_t^{Ma} = \tau_t^{Mb} = \frac{\rho T_t}{2c}. \tag{7}$$

Lemma 3. *The technology level set by a software monopolist increases for higher payoff, ρ, and a falling cost, c, but is independent of the degree of substitutability, β.*

Thus, given that an OSS developer can be fairly assumed to have a lower cost c, an OSS monopolist will provide a higher technology level than a for-profit monopolist. Yet, if the for-profit monopolist has a the higher payoff ρ compared to the payoff for the OSS developer, than the reverse conclusion applies.

Comparing (6) to (7) leads to the following proposition.

Proposition 1. *Given a global technology level T_t that can be expected to remain constant, each firm $i = a, b$ in a software duopoly sets a technology level τ_t^i for its respective software product that exceeds the technology level set by a software monopolist, τ_t^{Mi}.*

Proposition 1 states that a monopolist will choose a lower technology level compared to a software duopoly. The monopolist, in contrast to the duopoly, does take into account the externality that a high technology level in one software segment has on the dissemination of other software products in his portfolio. Proposition 1 carries an important message concerning claims that competition can be harmful in technology and research intensive industries. If firms do indeed compete in technology then competition – moving from a monopoly situation to a duopoly situation – enhances the technological level. This beneficial impact of entry occurs no matter if the entry takes place in a commercial software market or in an existing OSS market, i.e. the case of forking of an OSS project.

2.4 Technology shocks

We now examine how the above two settings react to an external shock in technology. Assume that time t is divided into two time periods $(1, 2)$ by an unexpected technological shock $\Delta T > 0$, such that $T_2 = T_1 + \Delta T$, and accordingly real technology levels of existing software are deteriorated. The process of adjusting the technology level $\tau_2^i > \tau_1^i$ of an existing software product commands a cost f. In particular, f captures the cost of issuing a new release of the software, development costs etc.

A software duopoly has to weigh the gain from adjusting the technology level $\tau_2^i > \tau_1^i$ relative to the cost f.

If both firms stick to the old technology levels given in (6), then their respective payoffs in period 2, only shown for firm a, are:

$$\bar{g}_2^a = \frac{\rho^2(\alpha + 2\beta)T_1\alpha\left(T_1\alpha + 2(\alpha + \beta)\Delta T\right)}{8c(\alpha + \beta)^2(T_1 + \Delta T)^2}\,. \tag{8}$$

Notice that $\bar{g}_2^a$ is larger than the payoff from both firms adjusting, i.e. setting the Nash equilibrium technology level. This can be seen by inserting the technology levels given by (6) into (3) to arrive at payoffs $g_2^a = g_1^a = \frac{\rho^2(\alpha+2\beta)\alpha^2}{8c(\alpha+\beta)^2}$. The reason is that not adjusting the technology levels brings the two firms (except for very large technology shocks, ΔT) closer to the cooperative equilibrium, i.e. the real technology levels set by the monopolist. Yet, since firms behave non-cooperatively, non-adjusting is not an equilibrium. Each firm has an incentive to defect and adjust its own technology level. Doing so, while assuming that the other firm sticks to the old technology level τ_1^b given in (6) and using the maximising technology level implied by (5), commands payoff:

$$\hat{g}_2^a = \frac{\rho^2(\alpha + 2\beta)\left(T_1\alpha + (\alpha + \beta)\Delta T\right)^2}{8c(\alpha + \beta)^2(T_1 + \Delta T)^2}\,. \tag{9}$$

Then, as long as $\Delta g^a = \hat{g}_2^a - \bar{g}_2^a > f$, and likewise for firm b, the duopoly will adjust to the technology shock. Inserting the values from the formula above gives the following condition for the adjustment of a software duopoly to a positive technological shock, ΔT:

$$\Delta g^D = \Delta g^a = \Delta g^b = \frac{\rho^2(\alpha + 2\beta)\Delta T^2}{8c(T_1 + \Delta T)^2} \;>\; f\,. \tag{10}$$

In order for this adjustment to be feasible, we also have the necessary condition that the actually realised Nash payoffs suffice to cover the costs f, in particular $g_2^a = g_2^b = \frac{\rho^2(\alpha+2\beta)\alpha^2}{8c(\alpha+\beta)^2} > f$. This means that the duopoly firms

are able to pay the adjustment costs f, needed to achieve the technological innovation of their software, out of their Nash profits.

From (10) we have the following result:

Lemma 4. *A software duopoly will be more likely to adjust to a global technology shock, ΔT, the larger its payoff, ρ, the lower its costs c, the lower the adjustment costs, f, and the closer the two products, β, are substitutes.*

Thus an OSS duopoly, which can be assumed to have lower costs c and adjustment (release) costs f, than a for-profit software duopoly, will more readily (that is, more frequently) adjust to global technological developments, and hence track global developments more closely.

Compare this to the incentives for a software monopolist. The monopolist – considering market A only – achieves when adjusting, i.e. setting $\tau_2^{Ma} = \frac{\rho(T_1+\Delta T)}{2c}$ from (7), payoff:

$$g_2^{Ma} = \frac{\rho^2 \alpha}{8c}\,. \tag{11}$$

Not adjusting, i.e. sticking to the old technology level ($\tau_2^{Ma} = \tau_1^{Ma} = \frac{\rho T_1}{2c}$), commands payoff:

$$\bar{g}_2^{Ma} = \frac{\rho^2 \alpha (T_1+\Delta T) T_1}{8c(T_1+\Delta T)^2}\,. \tag{12}$$

Then, a software monopolist will adjust the technology of both software products as long as $\Delta g^{Ma} = g_2^{Ma} - \bar{g}_2^{Ma} > f$. Setting in the values from above gives the following condition for the adjustment of the software monopolist to a technology shock:

$$\Delta g^M = \Delta g^{Ma} = \Delta g^{Mb} = \frac{\rho^2 \alpha \Delta T^2}{8c(T_1+\Delta T)^2} \;>\; f\,. \tag{13}$$

Parallel to Lemma 4 we have:

Lemma 5. *A software monopolist will be more likely to adjust to a global technology shock, ΔT, the larger its payoff, ρ, the lower its costs c and the lower the adjustment costs f are. Yet, the propensity to adjust the technological level is independent of the degree of substitutability β.*

This implies that an OSS monopolist, which can be assumed to feature both a lower c and f compared to a for-profit software monopolist can be expected to adjust its software product more frequently, thus tracking the global technological development more closely.

Comparing (10) and (13) a result on the willingness to adjust to technological developments under the two market scenarios can be derived. In particular, it becomes clear that for any $\beta > 0$, $\Delta g^D > \Delta g^M$. Thus the software duopoly will adjust for a wider range of f than a software monopolist.

Proposition 2. *Given a global technology shock $\Delta T > 0$ a software duopoly adjusts the technology levels of its products more readily than a software monopolist.*

Put differently, Proposition 2 states that for any situation where the monopolist will react to a positive technological shock, the duopoly will also adjust, but not vice versa.[11] The difference between the two market forms in their respective willingness to adjust to technological developments becomes more pronounced as β increases. Thus, as can be seen from (10), and as stated in Lemma 4, when the two software products are closer substitutes the willingness of the duopoly to adjust to a technological shock increases, while (13) is independent of β. Proposition 2 contains the intuitively compelling insight that, in a duopoly setting, not only do firms adjust the technology of their product in order to optimise their technology position with respect to the outside technological development, they also adjust in anticipation of the other firm's adjusting and thus stealing their market share. This effect matters more the closer substitutes the goods are, i.e. the closer competitors the two firms are.

Finally, from the above results we would expect to observe a higher frequency of technological adjustments (new releases), for OSS markets and for software markets that experience entry or forking. Thus such markets will be tracking the technological development more closely, i.e. adjusting to smaller shocks, while the monopolist stands still longer etc. The model further suggests that in mixed markets where one competitor is a commercial software enterprise and the other is an OSS developer, the chosen technological level and the adjustment speed to technological shocks are higher than in a pure commercial software duopoly, but lower than in a pure OSS duopoly. Thus, within the present framework, evaluated in terms of the technology level and progress, the pure OSS duopoly dominates all other market structures treated in the paper.

Furthermore, assuming comprehensively that a higher technological level and a faster adjustment to global technological developments are associated with higher user utility, and given that increasing the number of software

[11] This result is robust in relation to other forms of strategic interaction in the duopoly. In particular, examining the above situation for a Stackelberg equilibrium one can show that $\frac{\partial(\Delta g^D - \Delta g^M)}{\partial \beta} > 0$ and $\lim_{\beta \to +\infty} \frac{\Delta g^D}{\Delta g^M} = +\infty$.

firms meets no supply constraint in terms of programmer capacity, then our model implies that users benefit from entry of an OSS developer into a software market dominated by a monopolist. This holds for both cases: whether the monopolist is a commercial software enterprise or an OSS developer as well as for the case of forking and branching in OSS projects.

3 Conclusion

The paper analyses the influence of entry and competition by open source software (OSS) on innovation and progress in software markets. The best known example of such an event is the entry of Linux into the operating systems market. Incumbent commercial software producers claim that the technological progress in software will slow or even stop altogether as a consequence of OSS entry into former highly concentrated (monopolistic) markets. The empirical evidence available, although scarce, does not support this view. On the contrary, the data we present suggests that increasing competition in the software industry promotes if anything innovation. Based on these observations we set up a model of software competition where producers compete in technology rather than price or quantity. Within the model, the development decision of the firms regarding how to set the technology level of their software and the willingness to react to outside technological advances by upgrading their product is examined.

We find that the move from monopoly to duopoly always increases the technology level chosen by the enterprises. Also, the transition from a monopoly to a duopoly increases the willingness to adjust (the innovation speed) to global technological shocks. These results hold no matter if the incumbent firm is a commercial software producer (e.g. the entry of an OSS firm into the market of a for-profit monopolist) or an OSS developer (e.g. the case of forking within an ongoing OSS project). Furthermore, under the assumption that the development and innovation costs of OSS firms are lower than those of commercial firms, the model implies that a pure OSS duopoly dominates in terms of technology levels and rate of innovation monopolies (either commercial or OSS), pure commercial duopolies and mixed duopolies (e.g. one OSS firm and one for-profit firm).

To sum up, even though one might have to abandon the concepts of price or quantity competition when examining software markets – since software is an intangible good that can be replicated in unlimited quantities and is sold (by some producers) at price zero – this does not imply that competition is harmful. On the contrary, our approach shows that when one views software producers as firms that compete in technology rather than price or quantity,

then entry and competition will, via the strategic interaction of firms, still have a fundamentally positive impact on firms, increasing their willingness to innovate and heightening the overall technological level in the industry. Thus competition is still good, also when the product is software.

Appendix

Table A1: Release history of KDE

Version	Release Date
Beta 1	1997-10-20
Beta 2	1997-11-23
Beta 3	1998-02-01
Beta 4	1998-04-19
1.0	1998-07-12
1.1	1999-02-06
1.1.1	1999-05-05
1.1.2	1999-09-13
1.89	1999-12-15
1.90	2000-05-11
1.91	2000-06-14
1.92	2000-07-25
1.93	2000-08-23
OSS release*	2000-09-04
1.94	2000-09-15
2.0	2000-10-23
2.0.1	2000-12-05
2.1	2001-02-26
2.1.1	2001-03-27
2.2	2001-08-15
2.2.1	2001-09-19
2.2.2	2001-11-21
3.0	2002-04-03
3.0.1	2002-05-22
3.0.2	2002-07-02
3.0.3	2002-08-19
3.0.4	2002-10-09
3.0.5	2002-11-18
3.0.5a	2002-12-21
3.1	2003-01-28
3.1.1	2003-03-20
3.1.2	2003-05-19
3.1.3	2003-07-29
3.1.4	2003-09-16
3.2	2003-09-25

Source: www.kde.org

* KDE becomes open source software. The QT library – on which KDE is based – was released under GLP on September 4th, 2000. Only since this date can KDE be considered a proper open source program. We take this event to constitute the actual entry of KDE into the market of GNOME, which was OSS from the outset.

Table A2: Release history of GNOME

Version	**Release Date**
0.00	1997-04-29
0.10	1997-09-09
0.20	1997-09-17
0.30	1997-09-18
0.40	1997-09-19
0.50	1997-09-20
0.60	1997-09-30
0.70	1997-10-03
0.80	1997-10-10
0.90	1997-11-04
0.10	1997-12-08
0.11	1998-01-07
0.12	1998-01-21
0.13	1998-03-10
0.20	1998-06-10
0.25	1998-08-06
0.27	1998-08-14
1.0.9	1999-07-24
1.2.0	2000-04-25
1.4.0.1	2001-03-28
2.0.0	2002-06-10
2.0.1	2002-06-17
2.0.2	2002-06-25
2.0.6	2002-08-09
2.0.8	2002-09-05
2.1.0	2002-09-28
2.0.10	2002-11-15
2.1.2	2002-11-04
2.1.3	2002-11-28
2.1.4	2002-12-10
2.1.5	2002-12-16
2.1.90	2003-01-06
2.2.0.1	2003-02-04
2.2.1	2003-03-11
2.3.0	2003-04-10
2.3.1	2003-05-9
2.3.2	2003-05-22
2.3.3	2003-06-06
2.3.5	2003-08-01
2.3.6	2003-08-13
2.3.7	2003-08-30
2.3.90	2003-09-04
2.4.0	2003-09-10

Source: www.gnome.org

References

Bitzer, Jürgen (2001): High-Tech with Zero Development Costs: LINUX versus MICROSOFT, mimeo.

Bitzer, Jürgen and Philipp Schröder (2002): Bug-Fixing and Code-Writing: The Private Provision of Open Source Software, DIW Discussion Papers, No. 296.

Bloomberg (2001): Bloomberg News, retrieved from http://news.cnet.com, download date 15.04.2003.

Cassadesus-Masanell, Ramon and Pankaj Ghemawat (2003): Linux vs. Windows: Modelling Competition between Open-Source and Closed Software, mimeo.

Chaffin, Bryan (2001): Microsoft Sinks To New Low, in: The Mac Observer: The Back Page, http://www.macobserver.com, May 4th, 2001, download date 11.04.2003.

Chicago Sun-Times (2001): Interview with Steve Ballmer, in: Chicago Sunday Times, Cited in: Hildebrand, J.D.: Open source watch: Who needs Free Software Anyway, SDTimes, http://www.sdtimes.com, July 15, 2001, download date 11.04.2003.

Dalle, Jean-Michel and Nicolas Jullien (2002): Open-Source vs. Proprietary Software, mimeo.

Feller, J. and B. Fitzgerald (2002): Understanding Open Source Software Development, Amsterdam: Addison Wesley Longman.

Hars, A. and S. Ou (2002), Working for Free? Motivations for Participating in Open-Source Projects, *International Journal of Electronic Commerce*, Vol. 6 (3), pp. 25-39.

Johnson, J. P. (2001), Open Source Software: Private Provision of a Public Good, *Journal of Economics and Management Strategy*, Winter (2002), volume 11, number 4, pp. 637-662.

Lerner, J. and J. Tirole (2002), Some Simple Economics of Open Source, *Journal of Industrial Economics*, Vol. 50 (2), pp. 197-234.

Microsoft (2001): Microsoft PressPass Information for Journalists, http://www.microsoft.com/presspass, download date 11.04.2003.

Raymond, E. S. (2000a): Homesteading the Noosphere, Revision 1.22, 2000/08/24, first version 1998.

Raymond, E. S. (2000b): The Cathedral and the Bazaar, Revision 1.51, 2000/08/24, first version 1997.

Rosenberg, D. K. (2000): Open Source: The Unauthorized White Papers,

B&T; IDG Books Worldwide.

Rotemberg, Julio J. and Garth Saloner (1987): The Relative Rigidity on Monopoly Pricing, in: the American Economic Review, vol. 77, no. 5, p. 917-926.

Stallman, R. (1999), The GNU Operating System and the Free Software Movement, in: *Open Sources: Voices from the Open Source Revolution*, Chris DiBona, Sam Ockman, and Mark Stone (eds.), O'Reilly: Sebastopol, CA.

Torvalds, L. and D. Diamond (2001), *Just for Fun: The Story of an Accidental Revolutionary*, HarperBusiness.