Open Source Software Development, Innovation, and Coordination Costs

by

Jean-Philippe Bonardi, Richard Ivey School of Business, University of Western Ontario

and

Thierry Warin, Middlebury College

March, 2007

MIDDLEBURY COLLEGE ECONOMICS DISCUSSION PAPER NO. 07-01

DEPARTMENT OF ECONOMICS
MIDDLEBURY COLLEGE
MIDDLEBURY, VERMONT 05753

http://www.middlebury.edu/~econ

OPEN SOURCE SOFTWARE DEVELOPMENT, INNOVATION, AND COORDINATION COSTS

JEAN-PHILIPPE BONARDI

Richard Ivey School of Business
University of Western Ontario
London, Ontario, Canada, N6A 3K7
Email: jbonardi@ivey.uwo.ca

THIERRY WARIN

Middlebury College
Department of Economics
Vermont, 05753, USA
& Harvard University
Minda de Gunzburg CES
Email: twarin@middleburry.edu

Open source is often presented as a very promising governance structure for the development of software in the Internet world. One of its greatest advantages is that it enables and integrates the flow of innovation coming from many unrelated developers. We extend previous inquiries by showing that, due to information communication problems, this governance structure is in fact more efficient for the development of incremental innovations rather than radical innovations. Implications are drawn in terms of the future of the open source system, the economics of innovation and public policy.

(Open source software; radical innovation; governance structure)

ABSTRACT

Open source is often presented as a very promising governance structure for the development of software in the Internet world. One of its greatest advantages is that it enables and integrates the flow of innovation coming from many unrelated developers. We extend previous inquiries by showing that, due to information communication problems, this governance structure is in fact more efficient for the development of incremental innovations rather than radical innovations. Implications are drawn in terms of the future of the open source system, the economics of innovation and public policy.

(Open source software; radical innovation; governance structure)

INTRODUCTION

Open source software has recently been the object of considerable attention from the press, policy-makers and researchers (Blind & Edler, 2003). Crudely defined, open source software is software developed through an Internet community of volunteer developers, in which the source code – meaning the higher-level programming instructions that tell a computer what to perform – is made available to everyone. As such, the source code is non-proprietary and can be freely used, copied or distributed with or without modifications (Stallman, 1999). This system stands at odds with the one used by software companies since the 1970s, which have generally been choosing to withhold source codes to protect their proprietary software and reap commercial profits related to their innovations (Burgelman & Meza, 2001)..

Open source software – especially using popular examples such as the operating system Linux or the Apache Web server – has been the subject of several investigations in the economics and management literature (Krishnamurthy, 2003; Lerner & Tirole, 2001). A key driver of these investigations has been to understand why volunteer developers would decide to work for free to create public goods, something that has been seen by some as a challenge to standard economic principles, particularly regarding the relationships between individual incentives and innovation (Dalle & Jullien, 2003). As a response to this challenge, existing literature stresses the role of short-term benefits in terms of using the software, since these developers are

often professional programmers trying to fix bugs and adapt the software to their own needs (Johnson, 2002; von Hippel, 2001; von Hippel & Krogh, 2003). Other work focuses on delayed benefits related to the fact that developers' contributions to the most successful innovations are well identified and therefore create positive spill-over in terms of peer recognition and future career opportunities (Kollock 1999; Lerner & Tirole, 2002). Case studies confirm that these delayed benefits play a very important role in the open source community (Hertel, Niedner & Herrmann, 2003).

Evidence, however, suggests that a puzzle remains: if open source developers are mainly driven by these delayed benefits, one would expect outside developers to take risks and concentrate their efforts on real technological breakthroughs, i.e., software innovation with the potential to change the way people are doing things and therefore the likelihood of generating peer recognition and future career opportunities. In the rest of the document and following well-established literature in the economics of innovation, we will use the term "radical innovation" to characterize those breakthroughs, whereas more mundane market innovation will be called incremental (Abernathy & Clark, 1985; Damanpour, 1991; Henderson & Clark, 1990). The radical or incremental adjectives are not used in a market based definition where only the market would qualify ex post an innovation as radical or incremental (see the well-known example of Sony's Betacam). Here, we consider an ex ante definition that relies on the technological aspect

of the product, voluntarily putting aside some potential network economies that could undermine this definition.

However, empirical evidence shows that in fact the opposite is true: open source projects, even the most successful ones such as Linux, tend to be incremental innovations compared to that which was already available for users. It cannot be denied that open source software is generating innovations and that some of these innovations are creating value and developing market shares, but a significant number of them remain incremental innovations rather than radical ones. Why is that? What is the incentive mechanism one needs in order to develop a radical innovation?

The purpose of this paper is to explore this puzzle. We suggest that one idiosyncratic aspect of open source software development has been neglected by existing literature: the way information is exchanged between the project leader, i.e., the one at the origin of the innovation, and the developers. Because information is exchanged exclusively through e-mail messaging, actors function in an almost perfect information environment, in which it is difficult to determine who will make a strong commitment to develop the innovation. Hence, radical innovations are more difficult to plan and require a higher level of commitment by developers than would incremental innovation. The paper focuses only on the locus of innovation and type of innovation concomitantly with the possible coordination problem.

The first section of the paper explores the boundaries of open source as a governance structure. The second presents the model's basic set up. Section 3 then introduces communication problems, examines the open source development of an innovation in a situation of incomplete information and derives sequential equilibria. The last section concludes and discusses the implications of our results for the future development of open source as a governance structure for software innovation, as well as for public policy purposes.

OPEN SOURCE AS A GOVERNANCE STRUCTURE

Our starting point in this paper is to consider open source as a governance structure (Garzarelli, 2002; Williamson, 1985) with specific characteristics different from those of software companies (McCormack et al., 2001). Here, we underline the main aspects of this governance structure. The development process in an open source environment generally works in the following way: a project leader designs the first program, writes the basic lines of codes and makes them available to other developers (Raymond, 1999). The role of this leader is key to the open source system, as this is the first innovation attempt by the project leader. This initial attempt either attracts or does not attract further developers and triggers the software development process. The rest of the process relies on outside developers working on debugging and creating patches to improve the quality of the program, as well as writing new lines of code to add new features and applications (Kollock, 1999). In most cases, it is

also the leader who decides whether to include the debugs and patches proposed by the other developers or to discard them.

Outside developers who participate as volunteers in the project development are therefore the key factor in the open source system. As suggested earlier, existing economics literature has gone a long way in explaining why those developers might offer their services free of charge (Lerner & Tirole, 2002). Equally important, however, is the question of which projects outside developers will decide to participate in and how they will allocate their time and efforts among these projects. In effect, as noted by Raymond (1999), the pool of capable developers is limited, implying that, as the open source movement continues to develop, developers carefully consider various projects and make different levels of commitment to these projects. They can commit most of their time and effort to a project that they find particularly interesting and useful, or they can make only a weak commitment to many projects, thus making small contributions to numerous new versions of open source software. There seems to be a wide range of developers' behaviors regarding this strategy. Hertel et al. (2003), for instance, show that developers spend more time on Linux development when they feel that their contribution is highly important to the software itself.

A developer's decision whether or not to participate may also depend on the level of commitment made by others. In effect, before deciding to make a strong commitment to a very innovative project, a developer will probably

need to know whether others are ready to make the same kind of commitment. If they are not, then a very innovative project - one that is likely to demand the greatest amount of time, effort and creativity - will not hold its promises. Note that existing literature has stressed the role of an installed base of developers and has studied its effects on the probability of an open source project reaching completion (Johnson, 2002), therefore considering a network externality. Here, we concentrate on another type of externality, which is not based on the crude number of other developers, but rather on the level of commitment a developer perceives among others interested in the project.

Note also that this "commitment externality" affects not only potential developers, but also the project leader himself. Even though he might be convinced that his project has the potential to become a radical innovation, he might decide to invest only some of his time and effort into it because he knows he will need the support of other strongly committed developers to make it work. To some extent, one can even argue that a project leader tries to gauge the interest in a potential innovation by launching it on the Internet.

In many respects, the project leader plays a key role. While the leader's reputation may encourage people to commit to the project, along with the incentive discussions in the literature, there is another benefit of this leadership: the centralization of information. This centralization helps reduce

the asymmetries of information due to the horizontal organization of an open-source project compared to the vertical organization of a proprietary development. Indeed, an open source development relies on a network of developers around the world, rather than being centralized in one or just a few places, as is the case with the proprietary software industry.

Asymmetries of information are present in vertical organization as well as horizontal. But the degree of centralization of a vertical organization reduces these asymmetries. For the horizontal organization, asymmetries of information can be reduced in many ways. First, from a non-exhaustive list, the object-oriented nature of software reduces the asymmetries of information by clustering and focalizing developers on some specific pieces of software. Second, the use of collaborative tools – one being developed by open source developers is eGroupWare – introduces a degree of centralization almost equivalent to a proprietary software development firm. Although these two conditions are necessary, they are not sufficient to trigger a radical innovation. The third condition is important: the leadership. Indeed, the leader will impose the goals, the future developments, and, more importantly, the agenda. On this latter point, it is interesting to note that the website Sourceforge.net created an index and a ranking based on this index to measure the activeness of open source projects.

How then can the project leader and developers evaluate the level of commitment that others are ready to make? The answer is that, in an open

source community, they do this almost exclusively through e-mail messaging[1]. This process deserves attention. In effect, the open source model relies on modes of communication that are not as "information-rich" as others, such as face-to-face meetings: risks of miscommunication or misunderstanding are therefore potentially greater. In the next section, we create a model of open source software development that takes this characteristic into account.

MODEL SET-UP

PLAYERS

We represent a two-player game, $i = 1,2$. One player is the project leader and the other is the outside developer. The project leader would like the developer to take part in his project, while the developer must choose whether to make a strong or a weak commitment to the project.

At the beginning of each game, players make their decision based on what they know about the state of nature: $N = A, B$. A corresponds to a

[1] The relationships between the open source movement and the Internet are indeed very close. Some argue that it is the widespread use of the Internet that has caused the recent explosion of interest in this 20-year-old open source model. The Internet has allowed the open source movement to prosper because it has considerably lowered communication and collaboration costs for potential users and developers. The open source movement has existed at least since Richard Stallman's 1984 effort to develop the GNU software (Stallman, R. (1999). The GNU Operating System and the Free Software Movement. *Open Sources: Voices of Open Source Revolution*. C. Di Bona, S. Ockman and M. Stone. Sebastopol, O'Reilly and Associates.). However, only the rise of the Internet enabled the diffusion of open source efforts.

situation in which the open source software project is an incremental innovation, i.e., a minor improvement of a program that already exists. *B* corresponds to a situation in which the new software has the potential to become a radical innovation. This dichotomy might raise questions in the context of software. Software, in effect, is generally developed through an incremental process. So what constitutes a radical innovation in software? We consider here that a radical innovation is a project for which there is no pre-existing template software architecture (von Krogh et al., 2003). The architecture of a software characterizes the functionality of specific modules within the software and the interdependencies and interactions among these. In the case of a radical software innovation, therefore, the development process itself is a total discovery for developers.

In the game, it is assumed that the leader and the developer have similar pay-offs and cost structures. The leader has discovered an innovation and would like other developers to participate. But he must also determine whether or not to invest a lot of time and effort in the project, as must the developer. This decision will be based upon how one developer perceives the other developer's level of commitment. Both players wish to maximize their profits. Following Lerner & Tirole (2001), we assume that these profits are mainly delayed benefits, i.e., ways of creating strong signals about talents and innovativeness. Similarly, costs are primarily the opportunity costs of the

time and effort the developer spends contributing to the open source project. The objective functions can be represented by:

$$O_i(N) = \max \Pi(C_i) \tag{1}$$

where C_i represents the total cost of developer i in the state of nature A or B.

STRATEGIES

The leader and the developer each have two options: weak commitment (m) or strong commitment (M) to the project. Weak commitment means that the developer plans to address some issues but will also continue to work on many other projects in parallel. On the other hand, strong commitment means that the software developer plans to devote most or all his time and effort to the project. The total cost function is:

$$C_i = \begin{Bmatrix} \left(C_i^m(A); C_i^M(A)\right) \\ \left(C_i^m(B); C_i^M(B)\right), \text{ otherwise} \end{Bmatrix} \tag{2}$$

By assumption, we consider that:

- In state of nature A, both the leader and the developer are better off by making only a weak commitment to the project. The innovation is only incremental and is likely to be modularized, meaning that lots of other developers might be working on the project at the same time, all making incremental additions. Making a strong commitment to such a

project would probably be a waste of time and effort, since delayed benefits would not be high enough to justify that investment.

- In state of nature B, it would be rational for the two players to make a strong commitment to the development of the radical innovation. The opportunity cost is lower, and therefore we can write $C_i^M(B) < C_i^M(A)$.

Considering a Bertrand competition without capacity constraint, [2] the optimal-Pareto solution is thus:

$$\left\{\left(O_1^m, O_2^m\right)\middle|(A); \left(O_1^M, O_2^M\right)\middle|(B)\right\} \quad (3)$$

In such a configuration, payments are $O_i^m(A) > O_i^M(A)$ and $O_i^M(B) > O_i^m(B)$, and they prevent the prisoner's dilemma, as represented in Figure 1.

[2] It is one of the characteristics of the software industry.

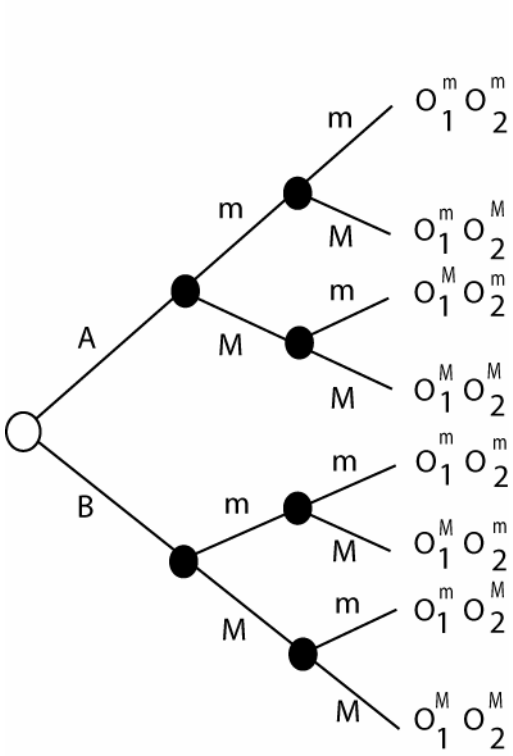

Figure 1. Decision tree.

COMMUNICATION IN THE CONTEXT OF AN OPEN SOURCE INNOVATION

In reality, the co-ordination mechanism of the commitment strategies of the two players in the open source project may be imperfect, as neither of the players can directly observe or monitor what the other does. Players communicate only through e-mails. Methods of communication between open source developers create, in the context of innovation, a situation of 'almost perfect information' (Rubinstein, 1989). Building on Rubinstein's approach, we model the interaction between the leader trying to attract developers and a developer. This interaction takes place within a context of incomplete information concerning the innovation potential of an open source project.

INCOMPLETE INFORMATION

In order to represent this "noise" in the co-ordination mechanism, we assume that the project leader has private information on the state of nature. In other words, he has a clear idea of the potential of the innovation. The project leader, then, freely passes this information to the other player, the potential developer. If the innovation is incremental, the leader simply posts the code on the Internet and does not send additional messages. On the other hand, if the innovation is radical, he also sends messages to advertise his innovation and encourage the outside developer to rally to it.

This transmission corresponds to the modeling of co-ordination. Each player must consider his or her own information, but also the information of the other player. Therefore, even when both players are certain of the project's innovation potential (either incremental or radical), if one actor is not sure how committed the other is to the project, that player can behave in ways that are contradictory to behaviors that would have been chosen based on this certainty.[3]

To begin the analysis, we assume that the most probable event is state of nature A, i.e., that the project is an incremental innovation. If B occurs, a message is sent from the leader to the developer claiming that his new project might become a radical innovation. The developer receives the message, understands that the project has the potential to be a radical innovation, and therefore sends a message back to the leader expressing interest. The project leader then responds with another confirmation of his expectations and commitment to the project, etc. This entire exchange is made necessary by potential failures of the transmission system: the information contained in the message sent by one of the players has a small probability of being lost or misunderstood by the other player, $q > 0$. In principle, this probability is small because hackers speak a common language and are all trained to program on Unix. The probability that a message still

[3] Here, both developers are volunteers and do not hide information strategically. We can introduce the idea of firms voluntarily willing to hide this information, but this is not something we consider in this paper.

circulates beyond a very large number of exchanges is thus *a priori* weak, but still exists and is not insignificant. Von Grogh et al. (2003) report, for instance, that in the case of the development of the open source project Freenet, the average number of e-mails needed before a joiner became a developer was 23.4.

Assume that, at some point, the communication ceases when the leader has sent a final message. This project leader in fact ends up in a situation of partial uncertainty: he knows the developer has expressed interest and even willingness to make a strong commitment to the development of the radical innovation. However, the project leader, not having received an e-mail from the developer, is left wondering whether the developer is still convinced that he, the project leader, is in fact strongly committed to the project. In other words, has the developer already started working hard on developing the radical innovation, in which case her last e-mail might have been lost? Or has she not responded because she did not receive the previous message from the project leader and therefore doubts that he is still committed to the development of the innovation?

SEQUENTIAL EQUILIBRIA

We consider the probability, q, that a message sent by one of the players may be lost or misunderstood. The game has an infinite horizon because of the back-and-forth transmission of messages. The procedure of

sending messages does not form part of the strategy: the real game (the open source software development itself) begins only when no further messages are exchanged between the two players, i.e., when both players have decided on the level of commitment they wish to make to the project.

Formally, we use the following notations to depict this situation:

- C_0, the beginning of the game ; the project leader discovers that the state of nature is either A or B with the probability distribution $(p,1-p)$ and $p>1/2$;
- C_t, the t[th] message (sent by the project leader if T is odd and by the outside developer if T is even);
- I_t, following sets of information:
    - I_A, the project leader discovered that the state of nature is A and sent no additional message to the developer,
    - I_0, the developer did not receive any message,
    - I_1, the project leader discovered that the state of nature is B and sent C_1 to the developer,
    - I_2, the developer received C_1, understood that the innovation had the potential to be a radical one and therefore sent C_2 to express willingness to make a commitment to its development;

- and more generally: I_{2t}, corresponds to the state of information of the developer when he sent C_{2t}, while I_{2t+1} is the project leader's information set.

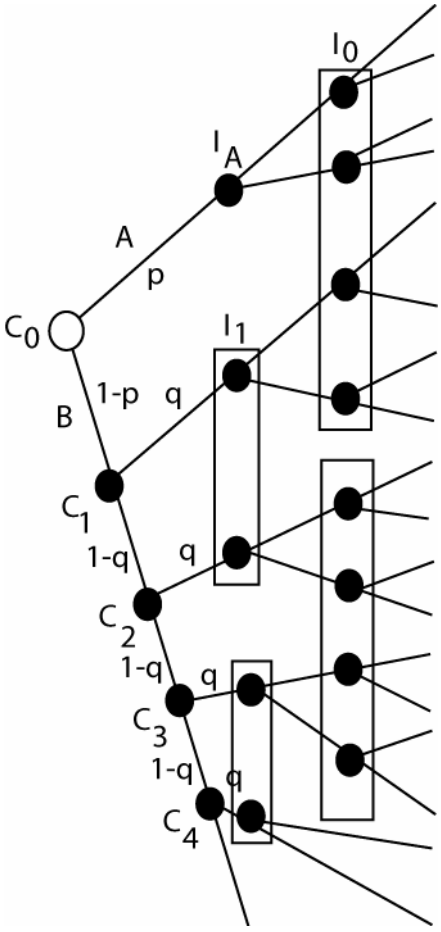

Figure 2. Developed form.

The fact that q>0 is not a trivial assumption. The interesting feature of this assumption and of the model that follows is that even when the organizational issues seem to be fixed by the technology, or the clustering of the whole projects in smaller projects (objects) – in other words, when the uncertainty seems to be resolved – the outcome may still be Pareto ineffective

due to the lack of leadership. For an even greater strength of the model, we put ourselves in a situation where a radical innovation is more likely than an incremental one: p>1/2.

Indeed, as soon as the developer receives a message regarding the new open source project, she knows that the state of nature is B. Thus, except I_A and I_0, the uncertainty is no longer due to the initial event, which is now known to both players, but rather to the state of information of the other player. For example, in I_2, the developer replied to the first message with C_2, and, as she did not receive any further messages, she does not know if the project leader is in I_1 (the project leader sent the first message C_1 but did not receive C_2) or in I_3 (the project leader received C_2 but did not send anything after that).

More generally, if the player's state of information is I_t, she does not know whether the other is informed of I_{t-1} or I_{t+1}. However, the probability of these two events taking place is not equal. In fact, we can show that, if a player sent a message C_t and did not receive a confirmation, there is more chance that C_t was lost rather than C_{t+1} confirmation did not arrive.

LEMMA 1. *If a player sent a message C_t and did not receive a response from the other player, it is more likely that C_t was lost rather than that C_{t+1} did not arrive.*

Proof. We calculate the conditional probabilities of I_{t-1} and I_{t+1} knowing I_t for any $t \geq 1$:

$$P\left(I_{t-1} \middle| I_t\right) = \frac{q}{\left(q + (1-q)q\right)}$$

and

$$P\left(I_{t+1} \middle| I_t\right) = \frac{(1-q)q}{\left(q + (1-q)q\right)}$$

thus:

$$\frac{P\left(I_{t-1} \middle| I_t\right)}{P\left(I_{t+1} \middle| I_t\right)} = \frac{1}{(1-q)} > 1.$$

Knowing I_t, a player knows that the other player is more likely to be in I_{t-1} than in I_{t+1}. □

The implication of Lemma 1 is that, when a player does not receive a message in which the other player confirms his strong commitment, the former thinks that the latter is in fact more likely to make a weak commitment rather than a strong one. If the developer did not receive a message, she thinks that it is more likely that the project leader plays as if the state of nature was A.

LEMMA 2. *The property of conditional optimality of a sequential equilibrium implies here that, whatever* $q > 0$ *and whatever the number of exchanged messages, co-ordination between the project leader and the developer cannot be applied with certainty.*

Proof. As $p > 1/2$, we have:

$$P\left(I_A \middle| I_0\right) = \frac{p}{p + (1-p)q} > P\left(I_1 \middle| I_0\right).$$

In other words, if the outside developer did not receive any messages, he or she thinks that it is more likely that the state of nature is A, rather than that the first message was lost. □

To obtain perfect co-ordination, the project leader must thus play m if A. As a consequence, the developer will also make a weak commitment within the context of an incremental innovation. The following proposition makes that clear.

PROPOSITION 1: *When the state of nature is A, the property of conditional optimality implies that the developer plays m.*

Proof. Let us determine a sequential equilibrium in which the project leader plays m if A. In this case:

In I_0, the developer minimizes its loss expectation, knowing that it will obtain:

$$\begin{cases} \min E\left(2 \middle| m\right) = P\left(I_A \middle| I_0\right) O_2^m\left(A\right) + P\left(I_1 \middle| I_0\right) O_2^m\left(B\right) \\ \min E\left(2 \middle| M\right) = P\left(I_A \middle| I_0\right) O_2^M\left(A\right) + P\left(I_1 \middle| I_0\right) O_2^M\left(B\right) \end{cases} \quad (4)$$

As $P\left(I_A \middle| I_0\right) > P\left(I_1 \middle| I_0\right)$ and $O_2^m\left(A\right) > O_2^M\left(A\right)$, the property of conditional optimality implies that the outside developer plays m. □

Similarly, even if the project has the potential to become a radical innovation, players will also make a weak commitment, leading to a suboptimal result.

PROPOSITION 2: *When the state of nature is B, the property of conditional optimality implies that both the project leader and the developer play m, even though the development of the innovation would require M.*

Proof. In I_1, the project leader knows B and knows that the developer plays m in I_0. Its expectations of conditional losses are then respectively:

$$\begin{cases} \min E(1|m) = P(I_0|I_1) O_1^m(B) + P(I_2|I_1) O_1^m(B) \\ \min E(1|m) = P(I_0|I_1) O_1^M(B) + P(I_2|I_1) O_1^M(B) \end{cases} \quad (5)$$

As $P(I_0|I_1) > P(I_2|I_1)$ and $O_1^m(B|m) > O_1^M(B|m)$, the property of conditional optimality implies again that the project leader chooses m.

By recurrence, the two players always choose m. □

This equilibrium is the fear of any leader, either in the proprietary software industry or the open source industry. If one wishes to extend the range of this conclusion, due to the differences in management in these two worlds, the open source development may be assumed to be less effective – or slower – at fixing issues due to asymmetries of information. If this holds true, and although the state of nature may be a radical innovation, open source developers in projects with a low commitment could a sub-optimal equilibrium.

CONCLUSION - DISCUSSION

The model developed here explores the choices made by leaders and outside developers in the context of an open source innovation (Blind & Edler, 2003). It shows that, having taken into account the uncertainty created by e-mail communication among developers, an open source governance structure creates incentives to under-invest in software that constitutes radical innovations. This provides an explanation for the puzzle underlined in the introduction of this paper: the most successful open source softwares tend to be incremental innovations, even though developers might gain greater benefits to their reputation by investing in radical ones. Our paper also explains a related empirical phenomenon, observed in several case studies of open source projects (von Krogh et al., 2003; Lerner and Tirole, 2002): open source developers tend to participate in many projects instead of focusing their efforts on the one they think is the most innovative (Burgelman & Meza, 2001)..

This paper therefore contributes to the existing literature on the economics of open source projects and software development, especially regarding what makes this open source governance structure more or less efficient depending on the situation. Short-term benefits of developers also being users (von Hippel & von Krogh, 2003), and delayed benefits in terms of reputation (Lerner & Tirole, 2002) have been identified by previous literature as being key criteria explaining how an open source governance structure

works and might be efficient in certain situations. Here we add another key criterion: the nature of the innovation, either radical or incremental.

Because of the coordination problems highlighted in the model, it is very difficult for a developer to know for sure that the project leader or other developers are ready to make a strong commitment to the development of the innovation. Note that firms generally do not face this kind of signaling problem. Firms, for example through expansive advertising or celebrity endorsements, can send a strong and credible signal to consumers, to suppliers or other corporate allies that they plan to make a strong commitment. As shown in this paper, this type of signal is much more difficult to create for open source developers. How an open source governance structure might, in certain cases, be able to overcome this problem is an interesting topic for future research.

Following our insights here, one can represent the relative efficiency of different governance structures by looking at two key dimensions (Table 1). On a vertical axis, one can consider the number of potential users of the innovation among the community of developers. This dimension takes into account the short-term benefits extracted by users of the software, as identified by von Hippel (2001). The horizontal dimension takes into account the nature of the innovation, either incremental or radical, as considered in this paper. As highlighted by the two grey areas in Table 1, our paper, combined with the existing literature, suggests that the open source model is

probably the most efficient governance structure when there are many potential users in the community of developers and the innovation is incremental. On the other hand, a closed source governance structure might be more efficient if the innovation is radical and there are few potential users among developers. In this case, the co-ordination problems stressed by our model should be quite difficult to overcome, making a corporate actor more likely to develop the innovation.

	Incremental innovation	**Radical innovation**
Many potential users	Open source – developers	Mixed governance structure: Closed source first, then full open source
Few potential users	Mixed governance structure: Closed source and then open source, but with the firm staying highly involved	Closed source – a firm

Table 1: Efficient governance structures for software innovations

Our discussion here also points out a clear avenue for future research. In effect, two areas remain to be studied in depth: when the innovation is radical with many potential users-developers, and when the innovation is incremental with few potential users among developers. For these cases, we indicated in Table 1 that mixed governance structures might be efficient options. By mixed governance structures –also been called *hybrid strategies* (West, 2003)– we mean that the innovation would begin as a closed source initiative and would move later to an open source one. Many companies have already engaged in these kinds of mixed governance structures, the most famous ones being Netscape with Mozilla (Hamerly et al., 1999), Hewlett-Packard with e-speaking software, or even IBM with its WebSphere suite which includes Apache.

One interesting aspect here is that the involvement of firms in some sort of open source project can be a double-edged sword. On the one hand, the firm can help create stronger delayed incentives for developers, in terms of future career benefits, recognition or press coverage of the innovation. These might be important supports when the innovation is incremental but there are few potential users. By removing delayed benefits, more developers might become interested in a project, even though they are not direct users. In this case, the firm might keep a degree of control over the development of the software, for instance by sitting on some sort of open source governing council which monitors the process.

On the other hand, however, firm involvement may also create some disincentives for certain developers to participate. Microsoft's attempts to participate in open source development, for instance, have not generated great support from developers (see for instance Tieman, 2001). In cases where there are many potential users among developers but the innovation is radical, this is probably a point to consider. A firm can certainly help start the process, but its continued control, for instance through a governing council, might create disincentives for developers. Releasing the product freely and fully into an open source model might be a good option in this case. These are just speculations based on the model proposed in this paper. Research in the future should certainly clarify whether these directions truly matter for the determination of mixed governance structures in software development.

Our results also speak to the literature on the economics and management of innovation as a whole. Starting with Schumpeter (1950) and Arrow (1962), this literature has focused on whether incumbents or new entrants were more likely to innovate. Further studies have shown that this depends on the nature of the innovation. In effect, an innovation can be incremental (or radical) either in the economic sense (which is the one that has been considered in this paper), or in the organizational sense (Henderson & Clark, 1990; Henderson, 1993). Compared to proprietary software, open

source projects are clear radical innovations in the organizational sense (von Krogh et al., 2003). However, we suggested here that they are unlikely to deliver radical innovations in the economic sense. Regarding the question of who is more likely to innovate, we might therefore reconsider Henderson & Clark's matrix as shown in Table 2. In that context, open source projects would be more likely to generate innovations that are incremental in the economic sense, but which are based on a brand new organizational structure. The open source system certainly has the flexibility to generate those organizational innovations. On the other hand, proprietary incumbents would remain more likely to innovate when the innovation is incremental both in the economic and organizational sense, due to their existing capabilities and their redeployment. Finally, proprietary new entrants would be the most likely to propose innovations that are both radical in the economic and the organizational sense. In this latter case, the coordination problems highlighted in this paper might make the open source governance structure less efficient.

	Incremental – Economic	**Radical – Economic**
Incremental – Organizational	Proprietary incumbents more likely to innovate	Unclear
Radical – Organizational	Open source projects more likely to innovate	Proprietary new entrants more likely to innovate

Table 2: Who is most likely to innovate in the software industry?

This analysis has deep implications for competitive dynamics in the software industry. It suggests that open source projects are in fact more likely to compete head to head with incumbents rather than with new entrants, both mainly targeting incremental innovations in the economic sense. The public policy debate concerning copyrights, in which open source proponents argue fiercely against software companies, takes on a new dimension from this angle.

Our analysis also has implications for public policy. Open source proponents are at the origin of a large controversy, especially in the European Union, regarding software copyrights. This paper does not take sides in this debate, but it does add another perspective: that open source governance structure is not necessarily the best way to develop software - it depends on the innovation. For radical innovations, there are some incentives

for innovators to go for a more integrated and hierarchical structure, at least in the beginning. That is not to say that there should be copyrights to protect software innovations, but rather that it would be inefficient to necessarily look for a decentralized governance structure.

By the same token, our paper also has implications for antitrust. A lot has been written on the Microsoft trial, and much of what has been written suggests that the very existence of Microsoft impeded the natural evolution of the open source mode of development. Our model suggests that, even in a world where no outside barriers to the development of the open source mode exist, there might still be some reasons why, for certain specific innovations, a more integrated mode might be superior. It might even be that the more innovative the project, the greater the incentive for the project leader to choose the closed source model. This would not be related to willingness by this project leader to benefit from monopoly rents, but would be the most efficient way to cope with the great uncertainty highlighted in this paper.

REFERENCES

Abernathy, W., Clark, K.B. 1985. Mapping the Winds of Creative Destruction. *Research Policy.* 14: 3-22.

Arrow. K. 1962. Economic Welfare and the Allocation of Resources for Invention. In *The Rate and Direction of Inventive Activity.* Richard Nelson (Ed.), Princeton, NJ: Princeton University Press: 609-626.

Behlendorf, B. 1999. Open Source as a Business Strategy. *Open Sources: Voices of the Open Source Revolution.* C. Di Bona, S. Ockman and M. Stone. Sebastopol, O'Reilly and Associates.

Blind, K., Edler, J. 2003. Idiosyncrasies of the Software Development Process and their Relations to Software Patents: Theoretical Considerations and Empirical Evidence. *Netnomics.* 5: 71-96.

Burgelman, R., Meza, P. 2001. The Open Source Software Challenge in 2001. *Case, Stanford University, Graduate School of Business,* SM-85.

Dalle, J-M., Julien, N. 2003. 'Libre' Software: Turning Fads into Institutions? *Research Policy.* 32: 1-11.

Damanpour, F. 1991. Organizational Innovation: A Meta-Analysis of Effects of Determinants and Moderators. *Academy of Management Journal.* 34, 3: 555-590.

Fichman, R.G., Kemerer, C.F. 1997 The Assimilation of Software Process Innovations: An Organizational Learning Perspective. *Management Science.* 43, 10: 1345-1364.

Garzarelli, G. 2004. Open Source Software and the Economics of Organization. In *Markets, Information and Communication.* Pierre Garrouste and Jack Birner (Eds.). New York: Routledge. 47-62.

Hamerly, J., T. Paquin, et al. 1999. Freeing the Source: The Story of Mozilla. *Open Sources: Voices from the Open Source Revolution.* C. Di Bona, M. Stone and S. Ockman. Sebastopol, O'Reilly and Associates.

Henderson, R. 1993. Underinvestment and Incompetence as Responses to Radical Innovation: Evidence from the Photolithographic Alignment Equipment Industry. *Rand Journal of Economics.* 24, 2: 248-270.

Henderson, R., Clark, K.B. 1990. Architectural Innovation: The Reconfiguration of Existing Product Technologies and the Failure of the Established Firms. *Administrative Science Quarterly.* 35, 1: 9-30.

Hertel, G., Niedner, S., Herrman, S. 2003. Motivation of Software Developers in Open Source Projects: An Internet-Based Survey of Contributors to the Linux Kernel, *Research Policy*. 32: 1159-1177.

Johnson, J. 2002. Open Source software: Private Provision of a Public Good. *Journal of Economics and Management Strategy*. **11**(4): 637-662.

Lakhani, K., von Hippel, E. 2003. How Open Source Software Works: "Free" User-to-User Assistance. *Research Policy.* 32: 923-943.

Kollock, P. 1999. The Economies of Online Cooperation: Gifts and Public Goods in Cyberspace. *Communities in Cyberspace.* Smith and P. Kollock. London, Routledge.

Krishnamurthy, S. 2003. *Business Horizons.* September: 47-56.

Lerner, J., Tirole, J. 2002. Some Simple Economics of Open Source. *Journal of Industrial Economics* **L**(2): 197-234.

Lerner, J., Tirole, J. 2001. The Open source Movement: Key Research Questions. *European Economic Review.* 45: 819-826.

McCormack, A., Verganti, R., Iansiti. 2001. Developing Products on "Internet Time": The Anatomy of Flexible Development Process. *Management Science.* 47,1: 133-150.

Mokus, A., Fielding, R., Herbsleb, J. 2002. Two Cases Studies of Open Source Software Development: Apache and Mozilla. *ACM Transactions on Software Engineering and Methodology.* 11, 3: 1-38.

Oram, A. 2000. Gnutella and Freenet Represent True Technological Innovation. *http://www.openp2p.com/pub/a/20805/12/2000.*

Raymond, E. 1999. The Cathedral and the Bazaar.
http:/www.tuxedo.org/esr/writings/cathedral-bazaar/.

Rubinstein, A. 1989. The Electronic Mail Game: Strategic Behavior Under "Almost Common Knowledge". *American Economic Review* **79**(3): 385-391.

Schumpeter, J. 1950. *Capitalism, Socialism and Democracy.* New York: Harper.

Stallman, R. (1999). The GNU Operating System and the Free Software Movement. In *Open Sources: Voices of Open Source Revolution.* C. Di Bona, S. Ockman and M. Stone. Sebastopol, O'Reilly and Associates.

Tiemann, M. 2001. Decoding Microsoft's Open Source Argument. http://zdnet.com.com/2100-1107-529784.html

von Hippel, E. 1995. *The Sources of Innovation.* New York: Oxford University Press.

von Hippel, E. 2001. Open Source Shows me the Way: Innovation by and for Users - No Manufacturer Required! *Sloan Management Review*. 42, 4: 82-86.

von Hippel, E., von Krogh, G. 2003. Open Source Software and the Private-Collective Innovation Model: Issues for Organization Science. *Organization Science.* 14, 2: 209-233.

von Krogh, G., Spaeth, S., Lakhani, K.R. 2003. Community, Joining, and Specialization in Open Source Software Innovation: A Case Study. *Research Policy.* 32: 1217-1241.

West, J. 2003. How Open is Open Enough? Melding Proprietary and Open Source Platform Strategies. *Research Policy.* 32: 1259-1285.

Williamson, O. 1985. *The Economic Institutions of Capitalism.* New York: Free Press.