Lehrstuhl für Unternehmensführung und -politik

Universität Zürich

Working Paper Series

Working Paper No. 8

Reconciling investors and donators - The governance structure of open source

Egon Franck, Carola Jungwirth

Juni 2002

Reconciling investors and donators - The governance structure of open source

Egon Franck, Carola Jungwirth**

(Preliminary draft, comments are welcome.)

Abstract: Software developed and produced in open source projects has become an important competitor in the software industry. Since it can be downloaded for free and no wages are paid to developers, the open source endeavor seems to rest on voluntary contributions by hobbyists. In the discussion of this puzzle two basic patterns of argumentation stand out. In what we call investor approaches, emphasis is put on the fact that although no wages are paid to contributors, other pay-offs may turn their effort into a profitable investment. In what we call donator approaches the point is made that many people contribute to open source projects without expecting to ever receive any individual rewards.

We argue that the basic institutional innovation in open source has been the crafting of a governance structure, which enables investment without crowding out donations. The focus of the presented analysis lies on the specific institutional mechanisms, by which the open source governance structure achieves to reconcile the interests of investors and donators.

Keywords: Governance, production and organizations, software communities
JEL: D200, L86, L22

** University of Zurich, Chair of Strategic Management and Business Policy, Plattenstrasse 14, CH-8032 Zurich, egon.franck@ifbf.unizh.ch, carola.jungwirth@ifbf.unizh.ch.

1. Introduction

Software developed and produced in open source projects has become an important competitor in the software industry, widely thought of as the realm of genuine capitalistic firms like Microsoft, SAP and so on. Let numbers speak.

According to market analysts the operating system Linux has between seven and twenty-one million users in the whole world and is growing faster than any other competitor.[1] As of November 2001 the web server Apache counted for 61.88% of active servers across all domains.[2] Sendmail routes an estimated 75% of mails in the Internet. Its closest competitors, Software.Com, Microsoft Exchange and Lotus Notes, hold just 3 percent of the market.[3]

Since in all these cases the software can be downloaded for free and no wages are paid to developers, the whole open source-endeavor seems to rest on the voluntary contributions of hobbyists. At the beginning of the open source euphoria Bill Gates wrote: „Who can afford to do professional work for nothing? What hobbyist can put three man-years into programming, finding all bugs, documenting his product, and distribute for free?"[4] Meanwhile it has turned out that the alleged hobbyists in some cases came up with software that is more elegant, consistent and successful than its proprietary counterpart.

In the discussion of this puzzle two basic lines of argument clearly stand out. In what we call investor approach, emphasis is put on the fact that although no wages are paid to contributors, other pay-offs may turn the investment of labor into an open source project into a profitable decision. Most prominent in this context is the argument by Lerner and Tirole (2001) according to which contributions ultimately translate into individual reputation, which is either comforting in itself or may be used as a talent signal on secondary markets, e.g. the job market or the market for venture capital.

---

1 See http://www.idc.com, http://leb.net/hzo/ioscount. See for example also Dalle and Jullien (2001), p. 3.

2 http://www.netcraft.co.uk/Survey/

3 http://news.cnet.com/news/0-1003-200-327370.html?tag

4 Bill Gates 1976 in his "Open Letter to Hobbyists"; See Moody (2001), p. 2.

In what we call donator approach, authors stress that many people contribute to open source projects without expecting to ever receive any individual rewards.[5] Because it is not money - as in commercial software production - and not peer recognition or a marketable talent signal - as assumed in investor approaches - which motivate these contributors, it seems appropriate to speak about idealistic motivations.

In many cases the interactions between "activists" of the "investor party" and the "donator party" are highly ritualized and stereotypical. The latter find proof in open source that the time has come to bury good old homo oeconomicus and economics altogether. "Aficionados" of economics work hard to show that there is a consistent incentive structure in open source, which secures contributions among rational actors. If this holds, idealistic motivations are nice, but not necessary to grasp the phenomenon. Economics does not need to take care of them.

Needless to say, both positions are unproductive. Donative behavior has a long history before open source, which can be studied in charities and in the nonprofit sector. It has not rendered economics useless, even if the sources of donative behavior are not well understood in economics. If we accept as an empirical phenomenon that people donate due to idealistic motivations, the question still matters which kind of organizations they choose for their contributions. Economics would still be in the game if it were a question of rational choice to allocate donations flowing from idealistic motivation among different recipients. The theory of nonprofit organizations pioneered by Hansmann (1980) makes a strong point in support of this view.

The position of an "aficionado" of the homo oeconomicus is unattractive as well. It not only turns out to be a truly philosophical endeavor in which an ever-increasing body of empirical evidence has to be ignored.[6] But, "aficionados" overlook that there is still room for economics in a world in which some actors are ready to contribute to certain activities even if expected benefits do not exceed incurred costs.

---

5 See for example Rota, von Wartburg and Osterloh (2002).

6 See Frey and Osterloh (2000), Frey and Jegen (2001) as well as Fehr and Falk (2001).

Open source is an ideal application for this kind of "enhanced" economics. It is a well-established empirical fact that both groups, investors and donators, are substantially involved in successful open source projects.[7] Given this fact, a basic institutional innovation of open source projects must have been the crafting of a governance structure, which enables investment without crowding out donations. The classical capitalistic firm that remains the most frequent organization structure in software development, serves the investors well who contribute work or money against future streams of income. Yet, its cornerstone, the institution of residual claims, precludes that a credible commitment can be given not to turn the donations of volunteers into private profits. It therefore drives out donators for reasons of contract failure, as we know from the theory of nonprofits (Hansmann 1980).

The aim of our paper is to explain the basic elements in the governance structure of successful open source software projects. In particular, we will try to lay open the specific institutional mechanisms by which the open source governance structure achieves to reconcile the interests of investors and donators.

The remainder of this paper is organized as follows: In section two we analyze how coordination and motivation for investors is achieved in open source and in traditional software development. Section three focuses on a governance structure suited to attract donators. It turns out that the same institutional prerequisite, a specific licensing agreement for software, ultimately enables the design of a rather complex incentive structure for investors based on reputation, and at the same time serves as a device against contract failure in attracting donators. Hence, this licensing agreement plays a key role in reconciling investors and donators in open source. Section four gives a brief outlook on other factors, which influence the success of open source projects.

7 Looking at the success of classical firms in the software industry, like Microsoft or SAP, one may ask why there should be any need for an additional governance structure for investors. We will discuss this point in paragraph 2.2.2. Lakhani and von Hippel (2000) as well as Hertel, Niedner and Herrmann (2002) have made an empirical investigation of the worked out the motivation mix that leads to contributions in open source systems.

2. Coordinating and motivating investors in open source and traditional software development

Investors contribute to an open source project because the expected value of their future pay-offs exceeds the incurred costs. Although traditional and open source software development both attract investors in this sense, they employ different mechanisms to do so.

There are two levels on which the alternative approaches to software development can be further described. First, there is a more technical level defining the work process and the division of labor. Second, there is the level of analyzing the incentives, which are needed to make sure that the actors perform according to the assigned tasks. At this level we have to take into account that rational actors only play expected roles if they earn a net benefit from doing so. In the terminology introduced by Milgrom and Roberts (1992, pp.25) the first level is known as the coordination problem, whereas the second is called the motivation problem. Now let us turn to the coordination problem first and abstract from the question of incentives for a moment.

2.1 Coordination

Software like Linux, Apache or Sendmail is highly complex. Many different features work together.[8] Since different users have different preferences, they employ different combinations of features. If the number of features that may or may not be used together is M, there are 2^M use-combinations. Since program features interact, the quality of software strongly depends on the testing and debugging of every single combination of use. Obviously, the number of combinations of use that need to be tested and debugged grows exponentially with the number of independent features added to the program.

Consequently, commercial software providers limit the amount of features as much as possible. The utilization of "structured code" and "object oriented" regimes facilitates the detection of bugs by predefining interfaces. In spite of all these techniques aimed at handling

8 For this argumentation see Bessen (2001), pp. 1-5.

complexity, testing and debugging remain the main cost drivers in software development and production, accounting for more than 80% of total cost[9].

2.1.1 Two approaches to software development: Disclosure-feedback and secrecy-incorporation

Putting it simple, there are two basic approaches in software development, which have different consequences with regard to the testing and debugging of complex programs. There is the disclosure-feedback approach used in open source projects and the secrecy-incorporation approach used by traditional firms in the software industry.

The communication facilities provided by the internet and the fact that software is an information good are basic enablers of the disclosure-feedback approach. This point will not be discussed extensively here, but it seems likely that the coordination approach applied in open source projects has not penetrated many other fields precisely because they lack the equivalent of the infrastructure of the internet.

In the disclosure-feedback approach actors are expected to play as follows: As soon as an innovator makes an improvement to the program, he discloses his work and invites peer review by others. Users selecting their own use-application from the combination of features automatically test the software, report encountered bugs and eventually write patches, try them out and again disclose them. Patches are peer-reviewed and if accepted as improvement become part of the next release of the software. Releases are frequent and debugging is done for every incremental improvement of the software. Feedback would not be feasible if the source code of the program were not distributed with the program. Users would not be able to fix bugs without being able to work on the source code and modify it. Peers would not be able to review improvements and proposed patches and infer their quality without looking at the modifications at the source code level.

In the secrecy-incorporation approach used in traditional software development the process is different: Software developers write source code. But this code will not be disclosed to the

9 See Ibid., p. 5 with reference to papers by Cusumano (1991) and Cusumano and Selby (1995).

users of the software. Instead, compiling it into binary object code, which cannot be easily reverse-engineered, hides this source code. Ultimately, the object code is incorporated into specific software products that are sold to customers. Without being able to access the source code, customers have only limited possibilities to debug programs. They cannot do more than report instances of malfunction back to the seller using "normal language". More precisely, they cannot inspect the source code, write patches, nor make improvements on their own. In addition, they are not in the position to peer-review modifications of the source code proposed by others. Strong feedback remains restricted to the narrower domain of software developers within the firm.

2.1.2 Different effects of coordination: rapid debugging versus versioning

Remember that incentive issues are still not taken into account for the moment. If we could take for granted the existence of institutional regimes perfectly supporting each of the presented approaches, the disclosure-feedback alternative should be faster with regard to testing and debugging complex software. The connection between intensive user involvement and innovation has been analyzed by various authors[10] and shall not be the main point of interest here. Nevertheless, the arguments supporting faster debugging of complex software in the disclosure-feedback approach are fairly obvious. The same people who work with the software applying it to their specific tasks also test it, report bugs and eventually propose solutions that they have already tried out. Communication breakdowns typically arising between dispersed users and the developers in the secrecy-incorporation approach are less likely to occur in this setting. Frequent releases, which come with the disclosure of every innovation, encourage simultaneous engineering. Innovators get their work tested and debugged at every step of the development process before bugs can infect larger program partitions. Unlike the number of customers in the secrecy-incorporation approach, the number of users in the disclosure-feedback regime is of direct relevance to debugging. With the source code laid open and accessible to modifications and peer review, every user is a potential bug fixer. Attracting users means increasing debugging capacity. Or, as Raymond states less formally: "Given enough eyeballs all bugs are shallow."[11]

---

10 See von Hippel (2001) and Harhoff, Henkel and von Hippel (2000).

11 Raymond (2000a), p. 2.

Nevertheless, even if rapid debugging is an obvious effect of the coordination approach termed disclosure-feedback, deferred debugging on the other hand is not a coordination failure of the secrecy-incorporation approach. Consumers will not pay for an update until substantial improvements are accumulated to a new release worth being called the next version. In our further considerations we have to take into account that disclosure-feedback and secrecy-incorporation define different work processes and different tasks that need to be carried out. Obviously, the incentive structure supporting each of theses approaches must be different too.

2.2 Motivation

As a point of departure, a closer look at the incentives supporting the secrecy-incorporation approach turns out to be useful.

2.2.1 The motivation structure of the secrecy-incorporation approach: intellectual property rights, residual claims, wages

The incentive structure supporting the secrecy-incorporation approach rests on the construct of intellectual property rights regarding the program code. These are defined and enforced in different ways.[12] Firstly, there is the concept of trade secrecy. As outlined above, the source code of software containing the innovation is hidden by compilation in the form of a binary object code and incorporated into software products. Secondly, there are user licenses by which consumers commit themselves not to redistribute the software product. Thirdly, there is the institution of copyright, which prevents that commercial software may be duplicated. Fourthly, there are patents by which it can be excluded that competing products contain infringing ideas. Moreover, there are common law and economic law, which enable companies to draw up employee confidentiality agreements and non-disclosure agreements. Law courts and public agencies, e.g. maintaining patent databases, complete the institutional

12 See Bessen (2001), p. 3.

regime by which intellectual property rights are defined and enforced.[13] Apart from these legal devices Microsoft permanently improves technical aspects of its software distribution system in order to exclude free riding by consumers simply copying the software. Before working with the new Windows version XP users need a starter code which will be delivered via internet if proof can be given that the software has been regularly bought.[14]

Intellectual property rights allow the owner of the source code (for example Bill Gates) to sell software products to consumers and to solve motivation problems in software development by applying the well-known governance structure of the classical capitalistic firm. Contributors working in the development team get compensated for their efforts by contractually prearranged wages. In order to prevent "shirking in teams" a monitoring structure is established. Being the residual claimant entitled to the net earnings after paying for the other inputs, the top monitor has incentives not to shirk his duties, and therefore serious monitoring will be implemented from the top to the bottom. Monitoring includes observing input behavior, apportioning rewards, giving assignments and instructions, terminating contracts and so forth.[15]

To summarize, the relative strength of the Microsoft approach to software production lies in the preclusion of consumers' free riding and in preventing team members in the development process from shirking. Investors in the sense described above fall into two categories. There are the software developers supplying labor and programming skills. They can expect fair wages as employees because specialized monitors motivated by residual claims prevent team members to shirk at the expense of others. Profits are the compensation for this second type of investors, the monitors.

---

13 Taken together, intellectual property rights grant innovators (at least temporary) monopoly power. This is thought to be necessary because of the positive externalities associated with innovations. Without intellectual property rights competitors would be able to imitate the new knowledge without incurring the research and development costs of the innovator. Since the social value of creating knowledge exceeds its private value, incentives to innovate are strengthened by granting innovators intellectual property rights which allow them some form of exclusive exploitation of the knew knowledge.

14 For example see Ludsteck (2001).

15 See Alchian and Demsetz (1972).

2.2.2 Another game in open source: abandoning direct income but gaining reputation

Incentives aimed at supporting the disclosure-feedback approach need to make sure that contributors can only derive utility from publication of their innovation. Recognition among one's peers is the "natural" utility achievable through publication. This kind of reputation can be satisfactory in itself or be traded on secondary markets. Lerner and Tirole (2001) have made reputation gains the central element in their explanation of the economics of open source. Peer recognition may trigger immediate ego gratification or enhance the bargaining position with respect to future job offers, access to venture capital, or shares in commercial companies selling complementary services to open source programs (consulting, training, packaging etc.). Reputation is a valuable asset in the labor and other secondary markets if it is a valid signal for otherwise hidden characteristics of the supplier or his offer. We will return to this issue later. Obviously, the problem of free riding by consumers simply vanishes in this context. Every down-loader pays attention to the software and therefore contributes to the reputation of its developer. In addition, he is a future potential contributor and perhaps promoter of the software.

Yet, the problem of shirking among contributors does not vanish if programmers are seen as reputation investors instead of workers compensated by wages. As with monetary incentives in classical software production, reputation too must be awarded in relation to the magnitude of the contribution/innovation made by the single actor.

However, how can an assessment system for contributions, which makes sure that nobody earns reputation by shirking at the expense of others be installed in open source projects? Without intellectual property rights over the source code, the option of a specialized capitalistic monitor motivated by residual claims is not feasible. It follows that only peers who are themselves contributors can do the assessment. In order for such a peer review to function there are several prerequisites. Contributions must be openly accessible to peers, which is the case when the source code of a program is laid open and distributed with the software. Moreover, innovators must be able to mark their contributions in a way securing recognition through several releases of the program. History, maintainer and credit files, which are attached to programs, play an important role in this context.[16]

16 See Moon and Sproull (2000) and Raymond (2000b), pp. 7-8.

In addition to these rather technical things, peers must also have incentives to make fair assessments of the contributions of others. This seems to be the crucial point in the whole debate. A functional equivalent to residual claims is needed. We believe that this functional equivalent is provided by the option to build reputation levers in a voting community.

In open source development innovators with a good reputation may embark on subprojects involving the improvement of a certain application or partition of the software. If they are able to attract good contributors, their reputation is boosted by the success of the subproject. Project leaders are under permanent assessment of their contributors who are interested in building their own reputation. If the assessments received by project leaders are not convincing (e.g. the project leader promotes friends, suppresses good contributions), they vote with their feet and subscribe to other subprojects. The project leader loses his reputation lever. With the option to build reputation levers, innovators receive a functional equivalent to residual claims, which makes sure that they are motivated to properly assess the contributions of peers. In fact, open source communities often have a vertical structure with more than one layer of project and subproject leaders.[17] Innovators at the top level have the highest reputation lever. They profit from the contributions of many programmers, which make up for the success of the software, but they also have much to lose if their decisions do not convince their "voters" anymore.[18]

Reputation gained in open source development may signal various abilities. The greater the contributions of an actor acknowledged by his peers at the source code level are, the better he has performed in a genuine programming tournament. Employers on the labor market can take the gained peer reputation as an indicator for programming skills and human capital that are not directly observable. The bigger the reputation lever that an actor was able to build in attracting "voters" for his projects and activities is, the better he has performed in a leadership tournament. Venture capitalists and owners of capitalistic firms can take the gained reputation

17 Take Linux for example: Linus Torvalds is the project owner standing at the top. He is followed by his "Trusted Lieutenants" who are his link to credited maintainers. Maintainers care for one module of the whole program assessing user contributions and keeping interfaces. See Dafermos (2001) or Moon and Sproull (2000).

18 There is an equivalent to this "lever building" in science. Becoming the editor of a prestigious refereed journal boosts one's reputation. But, at the same time, if contributors do not perceive the editorial policy as fair, they may vote with their feet. The same holds true for the building of research teams, where the most talented young researchers can also select their "project leader" at several academic institutions. See for example Franck and Jungwirth (2001).

as an indicator for management and leadership abilities which otherwise are genuine experience goods. This means that it would take years of expensive "experiments" to find out if someone is a good leader. As expected, open source communities make all the generated reputation-relevant information readily available, so that it may be used for transactions on secondary markets.[19]

The better the peer review works in an open source project, the more reliable are the produced signals for outsiders. But, how can they know if the peer review works? In practice they infer from the success of the produced open source software. A good "filter" is attractive for good programmers and leaders who would lose most if they were rated as average, as it would be the case without the filter. A project attracting good programmers and leaders is more likely to be successful. Therefore, in a competitive environment, success of the open source software is linked to good "filtering".

Summarizing this section it can be said that the incentive structure suited to support the coordination requirements of the disclosure-feedback approach is built on a reputation game. The option to build reputation levers in a voting environment creates a functional equivalent to residual claims. However, the described reputation game has a crucial prerequisite.

2.2.3 Copyleft: The constitutional prerequisite of the open source game

The major threat for the players of the reputation game described so far is the existence of a third party able to cut down the stream of attention flowing to them as reward for their contribution. This threat materializes if the developed code or parts of it are taken private sometime in the future and get incorporated into commercial software products. Every attempt to hide source code by claiming property rights on it means damaging the "citation mechanism" by which skills and effort are accredited.

19 It is a system of files that makes reputation visible. The so-called history file refers to the inventors of a program, maintainer files demonstrate respect for hard working contributors, credit files list people whose contributions are new features of a program and so on. See Moon and Sproull (2000) and Raymond (2000b), pp. 7-8.

Traditional contractual devices fail as a safeguard against such "pirating". A team of developers working on an open source project may draw up an agreement, according to which they will never incorporate and hide the code in commercial products. But even if we abstract from issues of enforcement, the disclosed source code is easy prey for outsiders who never signed such agreement. And there will always be such outsiders, no matter how many developers signed the afore-mentioned agreement.

Therefore, a constitutional device is needed which precludes every possibility to claim property rights on any program written making use of the open source program or parts of it. A basic institutional innovation in this context has been the so-called GNU General Public License (GPL, known also as Copyleft).[20] Those who wish to modify and distribute software under the GPL have to agree to make the source code available. They cannot impose any licensing restrictions on others. All derived work must also be distributed under Copyleft. The GPL or similar licensing agreements "infect" the open source software with a "virus" that makes sure that any software derived from open source software will remain open source software. Therefore, running an open source project under the GPL or similar licensing agreements sends a credible commitment to the investors in reputation that nobody will be able to cut them off from the stream of attention generated by their contributions.

The poor success of firms, which disclosed the source code of their commercial software in an attempt to tap improvements by direct user involvement, is unsurprising in this context. Innovators driven by the reputation incentive have no reason to invest in a program controlled by owners. By definition owners have the ability to take the program private whenever they wish to do so, and thus destroy the "citation mechanism" that ultimately feeds on the signaling capacity of contributors or just boosts their ego.[21]

---

20 http://www.gnu.org/copyleft/gpl.html

21 The most prominent example here is the dull success of the Mozilla-project. In 1994 Netscape successfully lanced the Internet browser Navigator as a commercial closed source software project but rapidly lost market share when Microsoft bundled different programs to the "MS Office" package. Now the Internet browser Explorer was delivered automatically. To defend its position Netscape decided to open the source code of the Navigator, making a free software project out of it. However, Netscape failed to offer license conditions contributors were willing to accept. See Dalle and Jullien (2001), p.7. and Hecker (1999). See also http://www.mozilla.org/mozilla-at-one.html.

3. A governance structure for donators

The signaling explanation does not answer the entire puzzle of open source. If outsiders infer "filtering quality" and the validity of signals from the success of the open source software in the market place, then reputation incentives can only emerge from already running open source projects. But, how does an open source project gain enough momentum to be able to attract all the reputation-motivated contributors described in the last section? How is the gap bridged until the project can deliver the fame these people seek to achieve?

One explanation is, of course, that some investors in reputation may gamble on the future and contribute in anticipation of future success. However, the prospects of such a strategy will be much better if the open source project is likely to gain initial momentum from the contributions of actors motivated by other goals than reputation maximizing. Since income is not a feasible alternative in this case either, two questions arise. Which are those other goals? Why is the open source movement a viable structure to pursue such goals?

The first question will not be analyzed at length here. It is well known from the literature that many open source communities have very strong ideological superstructures. The ideas people seek to express in the open source movement range from freedom, sharing, unrestricted exchange of information, anti-capitalism or anti-imperialism to fighting Microsoft.[22] Idealistic motives driving people to contribute without expecting individual compensation in the form of money, fame or other advantages may not be all too well understood in economics. Nevertheless, they are an empirical fact known from many fields, the most obvious of them being the various charitable institutions. We do not intend to further inquire into the question why people donate.[23] What interests more is the following point: Given that such other goals exist, why should they be primarily pursued through the contribution to open source projects? If we abstract from some extreme cases like for example anti-capitalism, the ideas are not restrictive a priori as to the form of organization that should be selected as a vehicle. For example, Microsoft's dominance could also be fought by deliberately buying products from its competitors wherever possible. The answer to the

22 See for example Raymond (2000a).

23 See Hansmann (1980) for this kind of argumentation.

question why open source organization is the right vehicle to attract contributors with idealistic motivations is more intricate.

3.1 Contract failure in a public goods context and Copyleft as nondistribution constraint

Many of the goals propagated by open source communities have characteristics of public goods. Goods qualify as public goods, if it does not cost more to provide them to many persons than it does to provide them to one person, and if there is no way to prevent other persons from consuming them, once they have been provided to one person.

Even a rather "simple" motivation like breaking Microsoft's market power refers to a public good which may be called "consumer freedom". Once Microsoft has been restrained, there is no way of preventing others from enjoying "consumer freedom" and it does not cost more to restrain Microsoft for additional consumers. The same holds for other ideals expressed in the open source movement like "information freedom", "relief of information barriers" and so forth. Even anti-capitalism or anti-imperialism have to do with public goods. Once capitalism or imperialism are abolished, nobody can be excluded from the blessings and there is no additional cost to extend the blessings to more people.

Economic theory explains that every individual should contribute to the production of a public good an amount equal to the value he places upon it, if the good is to be provided at the optimal level. Because the amount an individual may contribute will be very small in relation to the total investment, the production of the public good will not be affected by a single contribution. This translates into individual incentives to free ride. If the production gets started, the individual will be able to consume anyway. If not, non-contribution avoids losses. If all individuals follow this calculation, the public good will not be supplied although the aggregate demand may be substantial.

Standard economic reasoning concludes that the private market is unable to provide public goods. If this is used to call for the state, a very interesting point elaborated by Hansmann (1980, pp. 848-851) is missed. It suffices that the free-rider psychology outlined above is not universal and that there are (some) people willing to contribute toward the production of public goods. Such contribution to the production of a public good can be termed as donative

behavior because it does not influence the provision of the good to the individual contributor. The contribution is a donation precisely because the contributor's own consumption of the good is not affected by it.

Again, the reasons why people donate will not be elaborated here.[24] Suffice it to say that donative behavior is an empirical fact, which can be observed in many contexts. Hansmann[25] has explained why, taken for granted that (some) people are willing to donate in public goods contexts, nonprofit forms of organization are best suited to tap such donative behavior. If we take the example of a listener-sponsored radio station, the mechanism requiring that it should be nonprofit is straightforward:

> *"The listener knows what quality of broadcast is being provided, but he does not know whether his contribution is being used to pay for it. There is no observable connection between the amount of the individual's contribution and the quality of broadcast. The virtue of the nonprofit form of organization is that it can provide some assurance that in fact such a connection exists."*[26]

A basic institutional feature of nonprofits is what Hansmann (1980) calls the nondistribution constraint. Profits can be made, but they may not be distributed to those in control of the organization. In contrast to capitalistic firms nonprofits do not have residual claimants. As with the listener-sponsored radio station cited above this is an important contractual device in many public goods contexts.

Contributors have no means to assess if their donation flows into the production of the desired public good. Taking advantage of asymmetric information the residual claimants of the organization receiving donations could easily increase profits at the expense of the production of the public good. Because they cannot credibly commit to restrain from doing so, capitalistic firms with residual claimants face "contract failure" in attracting donations. Nonprofits can be an answer to this contract failure since the nondistribution constraint supplements the discipline of the market by an additional safeguard, the legal commitment that those in charge cannot extract any profits. In abolishing residual claimants nonprofits

---

24 Such inquiry on the psychological foundations of economic behaviour outlines the boundaries of economic rationality. Norms of fairness and reciprocity play an important role in human behaviour as many empirical and experimental studies show. See Fehr and Falk (2001).

25 See Hansmann (1980), pp. 849-851

26 Ibid., p. 851

make sure that nobody has incentives to increase profits at the expense of the production of the public good. To opportunistically turn donations into profits would only pay for somebody who had the right to take out the residual.

People may be driven to donate by idealistic motivations. But, if they have the choice to whom to donate, empirical observation confirms that they will select organizations with a higher propensity to direct their contributions into the production of a desired public good.[27] Because the nondistribution constraint contributes to make organizations fraud-safe for donators, it is suited to attract people with idealistic motivations.

Yet, how do these explanations apply to open source software production? The communities developing, producing and using Linux, Apache or Sendmail are not incorporated in the same way as the more traditional nonprofits. Universities, the Salvation Army, The Red Cross, or CARE are distinct legal unities. Just as in for-profit firms there is a corporate charter defining the legal boundaries of these organizations. In contrast to this, the legal boundaries of open source software projects are blurred. In the absence of labor contracts, decisions to participate or to contribute are easily reversible. Taking into account the internet-based interaction within open source communities, the boundaries in time and space are also fading. Open source projects are truly global.[28] Compared to the traditional nonprofits, open source communities are rather virtual forms of organization. At first sight it is hard to see how a virtual entity without clear legal borders can credibly commit itself to the nondistribution constraint, which makes for the basic competitive advantage over for-profits in solving specific instances of "contract failure".

By running software production and development under the terms of the General Public License or comparable licensing procedures, virtual communities too can commit to nondistribution. Those who wish to modify and distribute software under the General Public License have to agree to make the source code available. They cannot impose any licensing restrictions on others. All derived work must also be distributed under Copyleft. Copyleft is viral: It "infects" every software making use of source code produced under Copyleft with the requirement that it should be covered under the Copyleft license as well.

---

27 See Ibid., Fama and Jensen (1983), and Steinberg (1993).

28 For example, the credits file for the March 2000 release of Linux shows contributors from more than 30 countries. See Moon and Sproull (2000).

Copyleft is an intricate institutional device. It does not preclude that someone will ever be able to earn private profits with open source software. Because open source software always incorporates contributions of donators, any kind of profits made with it will contain an element of commercialization of donative resources. This is the case in the whole service industry growing around open source software products. Firms engaged in consulting, packaging and training in the Linux world partly commercialize donative resources because they use Linux as an "ingredient", whatever they may do. Programmers selling their knowledge and reputation to this service industry build their human capital in the donative context of the open source project. In this sense they may sell skills and signals they partly acquired through donations.

The point is more subtle. Copyleft is a device against fraud and not against private profits. Donations, which have entirely flown into the production of the public good first, may then be second-used and commercialized in the sense described above. By eliminating property rights on the source code, Copyleft makes sure that nobody can turn donations into private profits before they have contributed to the production of the public good. In the eyes of people interested in "free exchange of information" or "anti-capitalism" or "unrestricted sharing" someone who hides and incorporates source code in a software product, which he then sells on the market, simply steals their donations. Their work no longer contributes to "free exchange of information" etc. when hidden in a product sold for money. In contrast to this, someone who sells his reputation and knowledge acquired working on the software does not undermine the public good, since the software stays open source and "unrestricted sharing" etc. simply go on. On the contrary, he may further contribute to the dissemination of the software with his skills and reputation, in this way even increasing the installed base for "free exchange of information" etc.

Copyleft is a basic institutional innovation in the governance structure of open source. It encourages donators who identify with the broader purposes of the open source project to contribute by providing them with additional assurance that all support flows entirely into the production of the desired public good first. Yet, Copyleft does not crowd out investors allowing for the second-use and commercialization of the innovations partly coming from the input of donative resources.

3.2 The role of investor motivation for attracting donators and vice versa

For the sake of simplicity the quality of the open source software can be taken as a proxy for all the public goods donators might want to sponsor in open source projects. Whatever the idealistic goals, from fighting Microsoft to free exchange of information, powerful, consistent, elegant etc. software is the means by which these goals can be achieved.

By proposing that donators chose the form of organization, which provides assurance that a connection between the individual's contribution and the quality of the open source software exists, we assume that they make a rational choice. Their motivation to donate may have idealistic sources, but once they have decided to donate, they are concerned to find the organization that turns the highest quality software out of their contribution.

It is consistent with such behavior that donators should also take interest in the governance structure for investors. Their preferred public good is best served if the project attracts talented and motivated investors. In this sense an open source project with a well-designed governance structure for investors at the same time makes the highest quality software out of donations. Seen from the perspective of donators, investors seem like simple "working bees", which help the donator's ideals come true.

From a more general standpoint the relationship is symbiotic, because the profits investors try to capture in the form of marketable skills, signals and services partly stem from a commercialization of donative resources. In this sense, investors are better off in an open source project that has the right governance structure to attract large numbers of donators.

Bringing people with different motivations, investors and donators into a symbiotic relationship is the key innovation in the governance structure of open source. Idealists profit from investors as "working bees" who enhance the quality of software and thereby serve the ideals without consideration. Investors are compensated in a way that respects the constraint that all contributions should be directed to the improvement of the software first. Donators have no incentive to object the commercialization of derivative goods and services partly built in a donative environment, since this does not harm the quality of the software.

Before an open source project generates derivative exploits (marketable signals, skills and services), which are suited to attract investors, it must gain considerable momentum. In particular, the rather complex governance structure for investors must be designed and implemented, which consumes time and energy. How can a project start at all, if considerable set up investments need to be made before signals and other derivative goods start to flow? The symbiotic nature of open source governance may contribute to the explanation of this puzzle. The basic governance-requirements for donators are much easier to meet. As soon as a project is licensed under GPL or comparable agreements, donators are fraud-safe and may embark on their idealistic mission. If the concept they work on becomes prominent enough, investors who gamble on its future find their way in. Step by step the governance structure is completed in order to attract further investment.

4. Outlook

The focus of this analysis has been on governance structures. As important as a symbiotic governance structure reconciling investors and donators may be, it is, of course, not the only factor influencing the success of an open source project. We will briefly point out at two other important success factors.

Market conditions matter because they influence the cost-benefit calculations of investors. The spreading of Linux in professional surroundings such as companies or public authorities increases the demand for skilled Linux programmers. Opportunity costs for managing a Linux subsystem during leisure time for free rise proportionally to the external wage offers. Once an open source project produces high quality software, which gains significant market share, donators might conclude that the desired public good is already produced and available. If for example the dominance of Microsoft is broken and everybody enjoys free choice of software, there is no need for further donations[29].

The program conception matters because it defines the potential market of the program and the entry-requirements of contributors in terms of skills and knowledge. Presumably, open source projects need a broad potentially installed base of users in order to have a chance to

29 See von Hippel (2001).

succeed. A program that is of general use like an operating system and that is developed with standardized programming tools, enables many users to find bugs and even to fix them. On the other hand, one might argue that it does not pay to invest time in a standardized product that users can easily buy from commercial suppliers[30]. From this make-or-buy-perspective, highly specialized programs with a strong emphasis on quality are more appropriate to be developed open source and accordingly to be made instead of bought.

In our analysis we have assumed that favorable market conditions and program conceptions exist, whatever they might look like. Without a governance structure economizing on the symbiotic relationship of investors and donators, open source will not work. Collecting donations whenever given, providing a tournament whenever asked for is the crucial element in the governance of open source.

30 See Kuan (2001).

References

Alchian, Armen A. and Harold Demsetz. 1972. "Production, Information Costs, and Economic Organization." The American Economic Review, 62, pp. 777-95.

Bessen, James. 2001. "Open Source Software: Free Provision of Complex Public Goods." Working Version 5/01.

Cusumano, Michael A. 1991. Japan's Software Factories: A Challenge to U.S. Management. New York: Oxford University Press.

Cusumano, Michael A. and Richard W. Selby. 1995. Microsoft Secrets: How the world's most powerful software company creates technology, shapes markets and manages people. New York: Simon and Schuster.

Dafermos, George N. 2001. "Management and Virtual Decentralised Networks: The Linux Project." First Monday, 6. http://www.firstmonday.dk/issues/issue6_11/dafermos/index.html.

Dalle, Jean-Michell and Nicolas Jullien. 2001. "'Libre' Software: Turning Fads Into Institutions?" Working Paper: Cachan/Brest Cedex.

Fama, Eugene F. and Michael C. Jensen. 1983. "Separation of Ownership and Control." Journal of Law & Economics, 26, pp. 301-25.

Fehr, Ernst and Armin Falk. 2001. "Psychological Foundations of Incentives." forthcoming in: European Economic Review.

Franck, Egon and Carola Jungwirth. 2001. "Warum "geniale Ideen" für wissenschaftlichen Erfolg nicht ausreichen." Arbeitspapier des Lehrstuhls für Unternehmensführung und -politik Nr. 1: Zürich.

Frey, Bruno S. and Reto Jegen. 2001. "Motivation Crowding Theory: A Survey of Empirical Evidence." Journal of Economic Surveys: forthcoming.

Frey, Bruno S. and Margit Osterloh. 2000. "Motivation, Knowledge Transfer, and Organizational Forms." Organization Science, 11, pp. 538-50.

Hansmann, Henry B. 1980. "The Role of Nonprofit Enterprise." Yale Law Journal, 89, pp. 835-901.

Harhoff, Dietmar, Joachim Henkel, and Eric von Hippel. 2000. "Profiting from voluntary information spillovers: How users benefit by freely revealing their innovations." MIT Sloan School of Management Working Paper 4125.

Hecker, Frank. 1999. "Mozilla at One: A Look Back and Ahead." http://www.mozilla.org/mozilla-at-one.html.

Hertel, Guido, Sven Niedner, and Stefanie Herrmann. 2002. "Motivation of Software Developers in Open Source Projects: An Internet-based Survey of Contributors to the Linux Kernel." University of Kiel Working Paper.

Kuan, Jennifer. 2001. "Open Source Software As Consumer Integration into Production." Working Paper Haas School of Business, University of California-Berkeley.

Lakhani, Karim and Eric von Hippel. 2000. "How Open Source software works: "Free" user-to-user assistance." MIT Sloan School of Management Working Paper 4117.

Lerner, Josh and Jean Tirole. 2001. "The Simple Economics of Open Source." Working Paper.

Ludsteck, Walter. 2001. "Microsoft: Von einem Monopol zum anderen." Sueddeutsche Zeitung, 23.10.2001.

Milgrom, Paul and John Roberts. 1992. Economics, Organisation and Management. New Jersey: Prentice-Hall, Inc.

Moody, Glyn. 2001. Rebel Code: The Inside Story of Linux and the Open Source Revolution. Cambridge, Massachusetts: Perseus Publishing.

Moon, Jae Yun and Lee Sproull. 2000. "Essence of Distributed Work: The Case of the Linux Kernel." Firstmonday, 5, http://www.firstmonday.dk/issues/issue5_11/moon/.

Raymond, Eric Steven. 2000a. "The Cathedral and the Bazar." http://www.tuxedo.org/~esr/writings/cathedral-bazaar/.

Raymond, Eric Steven. 2000b. "Homesteading the Noosphere." http://www.tuxedo.org/~esr/writings/cathedral-bazaar/homesteading/.

Rota, Sandra, Marc von Wartburg, and Margit Osterloh. 2002. "Trust and Commerce in Open Source - a Contradiction?" University of Zurich Working Paper.

Steinberg, Richard. 1993. ""The Role of Nonprofit Enterprise" in 1993: Hansmann Revisited." Nonprofit and Voluntary Sector Quarterly, 22, pp. 297-316.

von Hippel, Eric. 2001. "Innovation by User Communities: Learning from Open-Source Software." MIT Sloan Management Review, 42, pp. 82-86.