Available online at www.sciencedirect.com
ScienceDirect
Research Policy 36 (2007) 157–171
research policy
www.elsevier.com/locate/respol

# Open source software development—Just another case of collective invention?

Margit Osterloh [*,1], Sandra Rota [2]

*University of Zurich, Institute for Organization and Administrative Science, Plattenstrasse 14, CH-8032 Zurich, Switzerland*

Received 11 December 2005; received in revised form 1 June 2006; accepted 15 October 2006
Available online 17 January 2007

**Abstract**

Do open source software (OSS) projects represent a new innovation model? Under what conditions can it be employed in other contexts? "Collective invention" regimes usually ended when a dominant design emerged. This is not the case with OSS. Firstly, the OSS community developed the institutional innovation of OSS licenses enabling OSS software to survive as a common property. Secondly, these licenses are mainly enforced by pro-socially motivated contributors. We characterize the conditions under which OSS communities develop and sustain pro-social commitments. We point out the vulnerability of these conditions to developments in patent legislation.


*Keywords:* New innovaton models; Open source software; Collective invention; Copyleft; Intrinsic motivation

## 1. Introduction

Open source software projects are examples of an innovation model, which relies neither on the price system nor on formal hierarchies nor on alliance agreements. Does the production of open source software (OSS) represent a new model, and under what conditions can it be used in other contexts? OSS is a term for software published under licenses that do not give any private intellectual property rights to the developers. To a large extent, contributions to these projects are made by voluntary, non-paid members of the OSS community. These members invest their own private resources into a public good.

A look at history shows that OSS production seems to be similar to other examples in the past and present that can be summarized by the term "collective invention" (Allen, 1983). In the second half of the 19th century, iron-making companies in Britain's Cleveland district willingly shared their innovations in blast furnace design. Other examples include the enhancement of steam engine design after 1800 (Nuvolari, 2002) and the search for a dominant design in the flat panel display industry (Spencer, 2003). According to Von Hippel and von Krogh (2003), OSS represents a novel innovation model, which they call the "private-collective" model. It shares many similarities with the "collective invention" model. However, most collective invention regimes did not survive after the development of a dominant design

* Corresponding author. Tel.: +41 1 634 28 40; fax: +41 1 634 49 42.
*E-mail addresses:* osterloh@iou.unizh.ch (M. Osterloh), sandra.rota@iou.unizh.ch (S. Rota).
[1] Margit Osterloh is Professor of Business Administration and Organization Theory at the Institute for Organization and Administrative Science, University of Zurich.
[2] Sandra Rota is a Research Assistant at the chair of Margit Osterloh at the Institute for Organization and Administrative Science, University of Zurich. Tel.: +41 1 634 29 39; fax: +41 1 634 49 42.

0048-7333/$ – see front matter © 2007 Published by Elsevier B.V.
doi:10.1016/j.respol.2006.10.004

(e.g. Meyer, 2002), while OSS is still alive and well in such a phase of the innovation process.

This paper addresses two research questions: (1) Why did "collective invention" usually not survive after the development of a dominant design? (2) What are the conditions that enabled the OSS collective invention to do just that? Answering these questions can help determine under what conditions "collective invention" or the "private-collective" innovation model can exist and be employed in other contexts than OSS (see e.g. Benkler, 2002, 2004; Von Hippel and von Krogh, 2003).

In Section 2, we give a short overview of the main characteristics of OSS. In Section 3, we present three other examples of collective invention. In Section 4, we explore what the different examples have in common. We discuss why collective invention normally does not survive the whole innovation process. In Section 5, we compare these findings with the OSS innovation process. We argue that OSS licenses, which are copyright-based, enable the collective invention mode to survive in a copyrighted world. In order to gain a better understanding of how OSS licenses can help with obtaining that goal, we describe the mixture of selective benefits that coexist in OSS compared to other collective invention cases in Section 6. We then ask what conditions exist preventing OSS making the changeover from a "private-collective" innovation model to a "private investment" model. We argue that the characteristics of OSS that are often identified as crucial, namely to be (a) an information product, (b) a user innovation that is (c) the product of a highly modular design, cannot fully explain the continued survival of the collective invention model. But they enable what we call a low-cost situation, which facilitates contributions to public goods. We argue that only the interplay of low-cost situations, OSS licenses and conditions that foster and maintain pro-social intrinsic motivation can explain the success of the OSS innovation model. We conclude with some remarks concerning the vulnerability of this model to developments in patent legislation.

## 2. Characteristics of 'open source'

OSS is an example of a continued survival of the collective invention model that challenges conventional economic views that innovation is best supported by strong private intellectual property rights (e.g. Demsetz, 1967; North, 1981).[3] In OSS, programmers place their software at the free disposal of other users and developers. The software is available on the Internet and anybody who is interested can download it for free. Users are not only allowed to use the functionalities of the software, but they are also permitted to view the source code of the programs.[4] If they wish to change, debug or develop the code further, they are free to do so. They are even free to publish these amendments together with the original or the changed source code. In this way, some projects managed to attract communities of thousands of programmers, who jointly develop the source without having private property rights on their produce.

The term open source essentially describes software licenses that explicitly give users the above-mentioned rights.[5] Nowadays, there are a dazzling number of different types of OSS licenses. The most important differences refer to the extent to which they allow public property rights to be mixed with private property rights. One of the most far reaching is the GNU[6] General Public License (GPL). It compels every program that contains a free software component to be released in its entirety as free software. In contrast to the conventional copyright, this license is called "copyleft". It "infects" the OSS with a "virus" to enforce compliance to the copyleft. It thus ensures that any software derived from it will remain free software. Other licenses, like Apache's, allow programmers to use their modifications as private property and distribute them as proprietary products. But even the least restrictive OSS licenses ensure that what once enters the public domain cannot be re-appropriated as a private good.

The really puzzling feature about OSS is that these programs partly constitute a public good in the classical sense (Lerner and Tirole, 2002), characterized by (1) non-excludability and (2) non-rivalry in consumption,

[3] Of course, this does not mean that all individual OSS projects are successful. Since the subject of this paper is to understand the necessary conditions for the survival of the collective invention model beyond the development of a dominant design in general, we will not go into further detail of what determines the success of a particular project. For empirical examples see, e.g. Raymond (2001).

[4] Source code is the human readable form of software. To transform the code into a form that is readable for a computer, the source code has to be compiled into a binary version, i.e. a sequence of computer instructions consisting only of strings of ones and zeros. Usually, when buying proprietary software, one only gets the compiled, binary version of the program. The computer can read these instructions and deliver the functionality of the program, but for human beings it is basically impossible to read and understand. With open source software, one gets both the binary version and the source code of a program.

[5] The open source definition includes some other defining features that are not of great importance for the arguments in this paper. To view the full definition, see, e.g. O'Reilly and Associates (1999).

[6] GNU is an acronym for "GNU is not unix", see Section 5.

which might create a problem of underprovision. Once the source is published, nobody can be excluded from using it. In the case of open source, not only is there no rivalry in consumption but also additional users can generate positive external network effects. But it still needs to be explained why the problem of underprovision seems to have been overcome in many projects.

Von Hippel and von Krogh (2003) argue that OSS represents a novel innovation model they call the "private-collective" model. The distinguishing mark of this innovation model is that economic actors invest their private resources in order to produce a public good. The authors argue that this model does not fit with one of the two major models in innovation theory, which try to explain how incentives to invest in innovation can be furthered, i.e. the private investment model and the collective action model. The private investment model states that private investment in innovation is furthered if inventors can appropriate the returns on their investment. To this end, inventors will try to avoid knowledge spillovers as far as possible, and society may grant intellectual property rights to the inventors in the form of patents, copyrights, and trade secrets. The collective action model applies to the production of public goods, like, e.g., basic research. For this model to work, some central agent (e.g. a university) has to grant selective incentives, e.g. monetary subsidies or reputation (Olson, 1965). Careful selection of contributors is important here, so that only individuals who are sensitive to these selective incentives are chosen. Von Hippel and von Krogh (2003) show that OSS resides in the middle ground between these models of innovation. They call it the "private-collective" model. Programmers freely reveal their innovations without claiming any private intellectual property rights. There is no central agent handing out selective incentives nor is there any hope of monopoly rents due to innovation. Contribution has to be self-rewarding, i.e., only programmers will contribute whose utility is greater when contributing than when free-riding. In that way, OSS seems to combine the benefits of the private investment and the collective action model while avoiding their downsides.

Under what condition did the "private-collective" model of OSS production succeed? Most answers to this question point to three characteristics of software production that greatly facilitate this kind of joint innovation: (a) since software is an information product, innovation and production are essentially the same thing, (b) users are innovators, and (c) the modular architecture of software design enables a decentralized production.

### 2.1. Information products

In contrast to physical goods, with information products, there is no rivalry in consumption. Giving information away does not preclude consumption by the donator. This property of non-rivalry basically applies to all non-physical objects, including ideas, designs, etc. (Baldwin and Clark, 2003). Thus, it could be concluded that the "private-collective" innovation model of OSS could apply to all innovation projects since, at least prior to the production phase, all innovation is a non-physical good. The only restriction here is that contribution still has to be self-rewarding, i.e. the benefits of contribution must exceed the benefits of keeping an innovation secret (Von Hippel and von Krogh, 2003). However, this conclusion needs further elaboration. While innovation in itself is definitely a non-rivalrous good, the rents one hopes to generate with innovation are not. If an innovation is widely diffused, the innovator loses his monopoly position and his rents might well deteriorate.[7] These opportunity costs are the reason why a manufacturer will not usually share his insights with competitors. For user innovators, as we will show in the following, these opportunity costs are often much lower, making sharing more likely.

### 2.2. User innovations

Users can be a rich source of (especially incremental) innovation (Von Hippel, 1988). By working with products, users get to know them intimately, can detect their shortcomings and develop ideas how the products could be altered to better suit their needs. However, they are often not in a good position to market their innovations. Even the licensing option is often quite unattractive, due to high transaction costs of intellectual property management (Harhoff et al., 2003). This leaves user innovators with two options: either they keep the innovation secret in order to exploit it in-house, or they decide to disclose it freely. In the case of physical products, a manufacturer might choose to incorporate the innovation in his product, and end up offering the user innovator a better and cheaper solution than he could have developed in-house. In the case of pure information products, like software, other users might build on the innovation and, in turn, release improved versions of the product that might again benefit the original user innovator. In both cases, the user

[7] This is not always the case though. In Section 4, we will come back to circumstances where sharing innovations can even enhance the profits of a manufacturer innovator.

innovator needs to weigh these potential benefits against the costs of disclosure. These costs are the lower: (1) the less other users can make use of the revealed innovation because it is too innovator specific, and (2) the less competition exists between the user innovator and other potential adopters of the innovation (Harhoff et al., 2003).

### 2.3. Modularity

Modularity is a technical and organizational way to manage complexity (Langlois, 2002). A modular system consists of a complex of components or subsystems where interdependencies among modules are minimized (Narduzzo and Rossi, 2003; Sanchez and Mahoney, 1996). Modularity enables many developers to work simultaneously on a project, while at the same time keeping integration and coordination costs low.

While these three characteristics of software production are definitely most important in explaining the huge success of some OSS projects, a look at history shows that the "private-collective" model of innovation we observe in OSS isn't a new phenomenon. Similar modes of innovation could be observed in the past, which Allen (1983) coined as "collective inventions".

## 3. Examples of collective invention

In the following, we will briefly present three earlier cases of "collective invention" that are similar to the "private-collective" model. We will compare them to the OSS case. The first case analyzed by Allen (1983) took place in the second half of the 19th century. It deals with the development of blast furnace design between ironmaking companies in Britain's Cleveland district. The two other, more recent examples include the Homebrew computer club (Meyer, 2002) and the search for a dominant design in the flat panel display industry (Spencer, 2003). In the following sections, we will present these three examples and compare them with the OSS case. We will argue that there are some differences between these examples of "collective invention" and the "private-collective" model characterized by Von Hippel and von Krogh (2003). The different models can be reconciled if one takes into account the different stages of development of the respective technologies. We will conclude that the above-mentioned three characteristics of OSS are not fundamental conditions for the emergence of a new innovation model, but facilitate conditions for its continued survival.

### 3.1. Blast furnaces in Britain's Cleveland district (second half of the 19th century)

Collective invention, as Allen (1983) coined the term, describes situations in which economic actors willingly reveal their innovations to an interested public in order that others can learn and, in turn, develop these innovations further. Technologies are thereby developed by repeated interaction and feedback. Information is shared with the whole innovation system and not just with a select few. Thus, collective invention distinguishes itself fundamentally from strategic networks, like formally agreed-upon R&D cooperations, and other contractual agreements, like cross-licensing.

When looking at the development of blast furnace technology in Britain's Cleveland district in the second half of the 19th century, Allen discovered that the bulk of innovation could be attributed to experimental incremental changes in design by firms, whose results were then made available to other firms in the industry and potential entrants. Neither non-profit institutions like universities and government agencies, nor firms or individual inventors, which patented their R&D outcomes were predominant. He concluded that there was a fourth innovative institution, which he termed collective invention. It led to dramatic technological changes in blast furnace technology. The average height of furnaces increased from 50 to 80 f, and the temperature of the blast increased from 600 to 1400 °F. These changes led to a significant reduction in fuel consumption. Sharing took place through informal disclosure and publication in the engineering literature. In 1869, the Iron and Steel Institute was established, which served as a forum for the presentation and exchange of technical material.

According to Allen (1983), these firms did not engage in formal research efforts. At that time, the chemical and physical aspects of blast furnace technology were not clearly understood. Nobody could predict what changes in productivity could be expected from design changes. Progress had to take the form of experimental trial and error learning. By sharing their experiences, firms could multiply the number of trials they could learn from, thus enhancing overall technological progress. But why should an individual firm, acting in a competitive environment, share relevant information and thus reduce its competitive advantage?

Allen sees three possible factors enabling the emergence of such behavior. Firstly, it was difficult to keep the information private anyway. Engineers and other employees who were involved in building the original facilities moved to other plants and took their experi-

ences along with them. Since the developments merely consisted of slight changes of existing designs, they were not patentable. Secondly, entrepreneurs were engaged in some kind of competition to see who was attempting the most daring designs. They might have been willing to release information, even if this meant sacrificing some of their profits. Thirdly, most entrepreneurs were also owners of ore deposits in Cleveland. Enhancing the efficiency of ore refinement technology enhanced the value of this complementary asset. Thus, they were not only interested in the profitability of their own plant, but also in the overall efficiency of iron ore refinement in the region. All of these factors enabled the emergence of a mutually beneficial sharing regime (Meyer, 2003).

For Allen, the most important lesson to be learned from his study is that collective invention can, to some degree, substitute for private R&D investment. But he suspects that nowadays, with the increasing importance of private R&D, collective invention should be less important. However, as the following two cases and the example of OSS show, this phenomenon is still important.

### 3.2. The Homebrew computer club (1975–1986)

The Homebrew computer club was formed at Stanford University in 1975, after the invention of the microprocessor, but before the advent of the personal computer industry (Meyer, 2003). It was a meeting place for people who had become interested in building applications for microprocessors, thus exploring their potential. The group was open to anyone who was interested, and it published a free newsletter. At club meetings, members would present their latest developments and discuss views about design philosophy. It was a meeting ground for people who shared interests, and numerous companies were formed. Steve Wozniak was one of them. When he built his own computer, he brought it to one of the meetings and passed around blueprints of its design so that others could copy it. He called his computer an Apple. Later he, together with Steve Jobs and some other Homebrew members, would form the Apple Corporation.

With the emergence of marketable products, the atmosphere at the club changed. People who had formerly willingly shared information, and who had only been interested in learning from each other, were suddenly in the position of competitors who had company secrets to hide from each other. They stopped going to the meetings and the club slowly lost its appeal. It disbanded altogether in 1986.

### 3.3. The flat panel display industry (1969–1989)

The first technological breakthroughs in flat panel display technology took place in the late 1960s. But it took 20 years until one product design, the liquid crystal display, emerged as the dominant design for laptop computer applications, and large-scale manufacturing and commercialization began. Spencer (2003) studied the knowledge sharing strategies of firms in the flat panel display industry in this pre-commercial phase between 1969 and 1989. She discovered that 80 of the 115 firms active in this industry published at least one scientific paper during this time. Spencer shows that firms that shared relevant knowledge with their global innovation system exhibit higher innovative performance than firms that did not share (measured by the value of their patent portfolio). Even though a causal relationship between the decision to share and innovative performance cannot be established, her data suggests that firms that employ the best scientists tend to have high numbers of publications and high innovative performance.

## 4. In search of common ground for the three examples of collective invention

In all three of these cases of collective invention, the actors involved were faced with a radical innovation, which is a time of high technological uncertainty and experimental learning processes.

A radical innovation is a technological development whose potential is not yet fully understood. While incremental innovations introduce relatively minor changes to existing products, exploiting the potential of the established design, radical innovations are based on a different set of engineering and scientific principles and often open up whole new markets and potential applications (Dess and Beard, 1984). These periods of experimentation are brought to an end by the emergence of a dominant design (Abernathy and Utterback, 1978). A dominant design is characterized both by a set of core design concepts (embodied in the components) and by a product architecture that defines the way in which these components are integrated (Clark, 1985). After the emergence of the dominant design, the level of experimentation drops significantly and technological development takes place at the component level within the framework of a stable architecture (Abernathy and Utterback, 1978; Henderson and Clark, 1990).

Technologies typically evolve along so called technological trajectories (Dosi et al., 1988). Radical technological breakthroughs cause disruptions in these trajectories. During an initial ‘era of ferment’ (Anderson

and Tushman, 1990), economic actors experiment with the new technology and rivalry centers on differences in the fundamental designs of the products (Teece, 1987). This is the exploration phase of a technology in which the potential of a radical innovation is explored. Typically, only one of these different trajectories will emerge as the dominant design for any given end application (Spencer, 2003). After that, the nature of competition changes and focuses more on incremental refinements within one trajectory (Abernathy and Clark, 1985). The technology enters its exploitation phase. During the exploration phase, firms cannot rely on their own stock of competences, but have to tap into external sources of knowledge (Stringer, 2000). There are two reasons why this is the case. Firstly, cognitive distance is typically greater between firms than within firms. This variety of cognition is a prerequisite to create novelty and to explore the potential of the new technology (Nooteboom, 2000). Secondly, different firms' approaches are often dispersed across distinct technological trajectories (Spencer, 2003). Careful observation of other approaches reduces the risk of getting stuck on a trajectory that ends up not being selected as the dominant design.

While it is rather clear why firms can profit from external sources of knowledge, it is less evident why they should be willing to share their own knowledge. In the alliance literature, knowledge sharing across firm boundaries is often represented as a social dilemma (see, e.g. Larsson et al., 1998). While all partners in the alliance would fare best at a collective level if everybody shared their knowledge, they can fare even better at an individual level if they manage to appropriate their partners' knowledge without contributing their own. If all partners follow this opportunistic strategy, the alliance will not be able to achieve its learning goals. There are three factors that can explain why, during the exploration phase, firms can overcome this social dilemma: (1) great learning potential, (2) low opportunity costs in the pre-commercial phase and (3) selective benefits.

(1) The severity of the social dilemma of knowledge sharing is reduced if the knowledge shared creates great synergies, i.e. sharing creates new knowledge that goes beyond the sum of knowledge stocks exchanged (Loebbecke et al., 1998). This is the case when sharing generates great learning potentials. Members of the Homebrew computer club stressed that there was so much to learn about the new technology that sharing their developments was the only way to explore its potential (Meyer, 2003). Iron makers in Cleveland had to build huge blast furnaces in order to try out new design concepts. For them, sharing their experiences was the only way to validate their results and to figure out which trajectories were the most promising for further experimentation.

(2) The severity of the social dilemma is also reduced if the potential negative impacts are low. In the pre-competitive phase, losses from sharing tend to be low, since there are not yet any marketable products fighting for a share in the market. On the contrary, Spencer (2003) argues that, for a new technology to be successful, it is imperative that several players converge on the same trajectory, e.g. by agreeing on the same standards, so that complementary products can endorse the new technology. This is facilitated if firms openly share their developments. By doing this, they might, once the technology enters the commercial phase, end up with a smaller piece of the pie than if they had kept their developments secret. But there is a fair chance that the pie itself increases in size.

(3) Still, the question remains why firms do not choose to sit on the sidelines and simply observe what others are doing rather then actively share their own knowledge. This puzzle can be solved if one considers selective benefits that only accrue to parties actively engaged in knowledge sharing. If such selective benefits exist, a social dilemma can be transformed into a coordination game where several equilibria exist (Kollock, 1998; Sen, 1974). In all of the above examples of collective invention, such selective benefits could be identified. They are as diverse as the cases themselves and include reputation benefits, gains through value enhancement of complementary assets, or pure enthusiasm about the potential of a new technology (see, e.g. Allen, 1983; Meyer, 2003; Nuvolari, 2003). One selective benefit that applies to all three cases is the influence on one's innovation environment. In order to develop a market for a new technology, it can be important that several players follow the same technological trajectory. Every firm has an interest that its own trajectory emerges as the dominant design, so that it can exploit its experiential advantage. By sharing their knowledge, firms might be in the position to attract other players to their own trajectory (Spencer, 2003). Sharing knowledge reduces barriers to entry onto the particular technological trajectory of a given firm, since it gives other players clues about what works, what problems they can expect and what methods might help in overcoming these problems. Additionally, if a firm is successful in this endeavor, it can optimally

profit from the developments of other firms that build on its efforts, since it induced others to invest their efforts in areas where it has a high absorptive capacity.

These three factors together (great learning potential, low losses from sharing, selective benefits from shaping one's innovation environment) explain why collective invention is observed in the wake of a radical technological development.[8] However, once a technology enters the commercial phase of competition, one would expect the collective invention mode to break down (Meyer, 2003). After the emergence of a dominant technological trajectory, competition shifts towards differentiation within a given dominant design, as companies fight for a share in the market (Teece, 1987). Appropriation concerns start to dominate; exploration of the potential of a technology is replaced by exploitation of its commercial value.

## 5. Is open source simply another case of collective invention?

OSS development is often described as the most global and successful case of collective invention (Meyer, 2003, Nuvolari, 2002) or the "private-collective" model (Von Hippel and von Krogh, 2003). However, most OSS projects do not fit into the conditions analyzed that make a collective invention regime likely: once a technology enters the exploitation phase, sharing knowledge promises less potential for acquiring new knowledge and for influencing the environment. Rather, the negative impacts of knowledge sharing increase, in particular rivalry about the appropriation of rents earned with the product. Sharing is no longer self-rewarding. Many areas of software development, especially end-user applications, however, are clearly in the exploitation phase. As Narduzzo and Rossi (2003) show, most OSS projects refrain from trying out new architectural designs. On the whole, they adopt already existing architectures. Also, the characteristics of user innovation and modularity clearly show that OSS has outgrown its exploratory phase. User innovation is a characteristic of the exploitation phase of a technology, where innovations are incremental, not radical. Modularity can only be achieved if a technology is well understood (Ethiraj et al., 2004; Langlois, 2002). The question arises why, in the case of OSS, a collective invention regime could survive the transition from exploration to exploitation. To answer this question, let us give a short account of the history of software development.

In the early days of computer programming in the 1960s and 1970s, commercial software was a rarity and was usually sold together with computer hardware. Users developed their own software, or hired programmers to write a particular program for a particular purpose. Most software was developed by scientists and engineers in academic and corporate laboratories. Since these programs were not intended for sale, but were rather working tools, developers freely exchanged codes they had written (Bessen, 2002; Narduzzo and Rossi, 2003; Von Hippel and von Krogh, 2003). This was facilitated by the establishment of the ARPANET in 1969, and later on by the Internet.

In that phase, which lasted till the beginning of the 1980s, major parts of software development have been, on the whole, really just another case of collective invention, and share astounding similarities with the cases discussed above. During the 1980s, however, three developments can be observed, which endangered this process. Firstly, the increasing dissemination of personal computers caused a growing separation between software users and developers. Secondly, copyright laws were extended on software, which made commercial uses more attractive. And thirdly, several platforms established themselves as some kind of dominant designs, which for the first time made larger, coherent software markets possible. Proprietary development and marketing started to dominate the process of collective invention. This also happened in the case of Unix.

Unix is a highly modular, scalable and portable platform. Its architecture is massively decomposed into independent modules that 'do one thing and do it well'. The history of its development goes back to the late 1960s, when AT&T Bell Labs were responsible for most of the coordination. AT&T freely distributed Unix to universities and commercial firms. The releases included the source code of the software. This enabled users to contribute to its development. Developers at MIT and Berkeley were especially active contributors, but commercial firms also released their own diverging versions of Unix, optimized for their own computer architectures (www.unix.org/what_is_unix/history_timeline.html). In 1983, AT&T decided to end the confusion caused by all the different versions, and combined various versions developed at other universities and companies into Unix System V Release 1, the first supported release (http://www.en.wikipedia.org/UNIX). This new commercial Unix release, however, no longer included the source code and was sold for 50,000$ per license.

[8] For further examples of such cases see, e.g. Meyer (2003) and Nuvolari (2002).

Other software companies were formed to distribute their own operating systems based on the Unix architecture, among them Santa Cruz operation (SCO) and Sun Microsystems (http://www.en.wikipedia.org/wiki/UNIX). Appropriation concerns came to the fore and the collective invention regime received a severe jolt.

As a result, many programmers saw themselves excluded from the development of a product they had helped to create. In 1984, Richard Stallman reacted by starting a new project called GNU, an acronym for 'GNU is Not Unix' (Narduzzo and Rossi, 2003). The aim was to develop a free alternative to the existing commercial and proprietary Unix versions. To prevent GNU from suffering the same fate as Unix, Stallman developed the GNU Public License (also known as "copyleft"). This license states that people are free to read, use and change the software, as long as the modified code contains the same freedom rights as the original code. By doing this, Stallman used existing copyright laws in a most unusual way: rather than claiming property rights, he used copyright to prevent anybody from appropriating the code. The free software movement was born. The term "open source" was introduced in 1998 at a meeting of free software activists, since they felt that the older term "free software" inspired thoughts of "gratis" rather than freedom of opinion (O'Reilly and Associates, 1999). Thus, OSS production can be seen as a movement to keep collective invention alive and well, even when the technology has reached its commercial phase. It seems that OSS licenses are an ingenious institutional innovation that mark the main difference to the other cases of collective invention. They are the root of a new innovation model, which goes beyond traditional forms of markets, hierarchies and contractual strategic alliances. Benkler (2004) calls it a social sharing system. To understand how, and under which conditions, this could be achieved, we now have to take a closer look at the motivational aspects of contribution to OSS projects.

## 6. Motivations to contribute to open source software

There is a whole range of different motives that act as selective incentives for contribution. If the contribution to OSS is self-rewarding, the social dilemma situation of knowledge-sharing is transformed into a coordination game, where defection is no longer the dominant solution. This is the case when the benefits exceed the costs. In Section 6.1, the co-existence of different selective benefits in OSS is described. In Section 6.2, we ask what differences there might be between motivations in OSS and other cases of collective inventions.

### 6.1. Selective benefits to contribute to open source software

The question of what selective benefits accrue to contributors of OSS projects, which are not accessible to free-riders, has been widely addressed by scholars interested in the phenomenon (e.g. Ghosh et al., 2002; Hars and Ou, 2002; Von Hippel and von Krogh, 2003). Most of the empirical studies in this field concentrate on individual contributors and question them about their motives to participate in OSS projects. It is, however, interesting to note that, although OSS was started by individual programmers as a countermovement to the commercial software world, the success of some projects in time attracted the attention of just this commercial world. In the following we first turn to the motives of individual contributors and then turn to the reasons why by now many commercial firms participate heavily, at least in some, OSS projects.

#### 6.1.1. Selective benefits that accrue to individual contributors

The selective benefits that accrue to individual contributors are often characterized by two ideal types of motivation, intrinsic and extrinsic motivation (Amabile, 1996; Deci and Ryan, 1985; Frey, 1997; Osterloh and Frey, 2000; Staw, 1975).[9] Extrinsic motivation works through indirect satisfaction of needs, most importantly through monetary compensation. Intrinsic motivation works through immediate satisfaction of needs. An activity is valued for its own sake and appears to be self-sustaining. The ideal incentive system for intrinsic motivation consists of the actual work content. Intrinsic motivation can be enjoyment-based or pro-social (Lindenberg, 2001). Enjoyment-based motivation refers to a satisfying flow of activity (e.g. Csikszentmihalyi, 1975), such as playing a game or reading a novel for purely personal pleasure. With pro-social motivation, the good of the community enters into the preferences of the individual, such as charitable giving (Frey and Meier, 2004), organizational citizenship behavior (e.g. Organ and Ryan, 1995) or voluntary adherence to rules, even if this is not in the person's own self-interest (Tyler and Blader, 2000).

[9] Intrinsic and extrinsic motivation are the two ends of a wide continuum (Deci and Ryan, 2000). We do not refer to the whole continuum, but adhere to the rough distinction, since it has been widely adopted in the empirical OSS literature.

As empirical work has discovered, programmers have intrinsic and extrinsic reasons for contributing to OSS (Gosh et al., 2002; Hars and Ou, 2002; Lakhani et al., 2002). Extrinsic motives include benefits gained from extended functionality and reputation. Intrinsic motives include benefits gained from enjoyment and pro-social motives.

*6.1.1.1. Benefits from extended functionality.* Contributors to OSS can gain extrinsic benefits as user innovators or lead users, who tailor the software to their own needs (Von Hippel, 1988; Von Hippel and von Krogh, 2003). They publish their amendments, because other users might work with the amendments of the code, maintain and develop them. Because the costs of publication on the Internet are low, it can pay off, even if the expectations for helpful comments from other users are relatively low.

*6.1.1.2. Reputation motives.* Contributors can make money indirectly by signaling their capabilities to the OSS community at large, which can then be turned into money through employment by a commercial software company or through easier access to venture capital. In OSS projects, it is argued that it is easier to get credited with a good reputation than in proprietary projects, thanks to the filing system that lists people who made contributions, and due to the public nature of mailing list archives (Lerner and Tirole, 2002; Moon and Sproull, 2000).

*6.1.1.3. Enjoyment.* There is a lot of evidence that, for a lot of programmers, the work itself is intrinsically rewarding, based on personal enjoyment. Many actors of the OSS community emphasize that the most important motives for participation are having fun, learning and the public display of one's capabilities. Writing or debugging software is perceived as a "flow experience" (Csikszentmihalyi, 1975). More than 70% of OSS developers report that they lose track of time while programming (Lakhani et al., 2002). As Raymond (2001) puts it: "We're proving not only that we can do better software, but that joy is an asset" (see also Torvalds, 1998).

*6.1.1.4. Pro-social motives.* The OSS community is often described as a gift-culture instead of an exchange culture (e.g. Raymond, 2001). OSS contributors report that they like the sense of "helping others" or "giving something back" to like-minded people (Faraj and Wasko, 2001). Contributors seem to reply to the entire group when answering a personal question (Wellman and Gulia, 1999). People who contribute for normative reasons produce a public good of two different orders. Firstly, they contribute to the functionality and quality of the programs (first order public good). Secondly, they are engaged in keeping the source code open (second order public good).[10]

### *6.1.2. Selective benefits that accrue to commercial firms*

Commercial firms more and more use the innovative power of the OSS innovation model. By now about half the development effort on OSS projects is performed by programmers at work, either for the firm itself, or at least with the knowledge of their supervisors (Lakhani and Wolf, 2005). Commercial firms are absolutely vital for the success of OSS because of two reasons. Firstly they contribute to OSS projects. Secondly, contrary to most individual hackers, they have an interest to provide even inexperienced consumer with reliable services and add-ons (Kogut and Metiu, 2000).

Some firms grew out of the OSS context and mainly base their business model on OSS products. Red Hat for instance sells support and services to Linux and other autonomous components. Other firms decided to open source projects which where up to then developed as proprietary products. Netscape was one of the first to open source a major product and caused an actual sensation when in 1998 it decided to publish the source code of its Netscape Communicator and started the project Mozilla. Others choose to contribute resources to existing OSS projects. IBM employs a whole bunch of Linux developers and makes its hardware compatible with it. Hewlett Packard sells hardware, like printers, and contributes code in the form of add-ons, like printer drivers, to make its products work with OSS. Henkel (2004) groups the motives of firms to contribute to OSS projects into five categories: Setting a standard and enabling compatibility; increasing demand for complementary goods and services; benefiting from external development support; signaling technical excellence and/or good "OSS citizenship"; and adapting existing OSS to the firm's needs.[11] Most business models that incorporate OSS profit from the wide diffusion of the respective software (Henkel, 2004). The market for these firms is extended because users can modify OSS to meet their specific needs which could not profitably be met by standardized products (Bessen, 2005; Franke and von Hippel, 2003). As a consequence, most firms that use OSS

[10] For first and second order public goods see the next section.

[11] For an overview of various business models see Markus et al. (2000).

mix proprietary and OSS solutions (Bonaccorsi et al., 2006).

To summarize, the co-existence of intrinsic and extrinsic motives and of individuals and firms is crucial to the success of OSS. On the one hand, intrinsically motivated individual contributors help projects to gain momentum, so that they become attractive for extrinsically motivated contributors (Franck and Jungwirth, 2003). On the other hand, extrinsically motivated contributors and commercial firms help to propel these programs into the mainstream.

### 6.2. What is special about open source software?

The mixture of extrinsic and intrinsic motivation, and of individual contributors and commercial firms discussed here might not be specific for OSS. Though we do not have empirical evidence, it might also exist in other cases of collective invention. Besides, there is considerable empirical evidence in other contexts that a lot of people contribute voluntarily to public goods or commons (e.g. Frey and Meier, 2004; Kollock, 1998; Ledyard, 1995; Ostrom, 1990). The question arises as to what is so special about OSS that the voluntary contribution to a public good continues after the emergence of the dominant design, when sharing of knowledge is no longer self-rewarding in the other cases. The answer lies in the fact that OSS is better able to solve the first and second order social dilemmas than the other projects.

#### 6.2.1. How the first order social dilemma is mitigated in open source projects by low-cost situations

The first order social dilemma is mitigated without a central authority when the selective benefits that accrue to contributors of a public good are self-rewarding. This is the case as long as the (extrinsic and intrinsic) benefits exceed the costs of contributing. Even among intrinsically motivated donators, martyrs and saints are in short supply. Donators are more willing to contribute if the private opportunity costs are not too high (Rose-Ackerman, 2001, p. 553). According to (North, 1990, p. 43), the demand curve for moral concerns slopes downwards. The more costly it gets, the less people contribute. On the other hand, if a low-cost situation exists, many people contribute small parts to the public good, so that the total amount of contributions rises considerably (Kirchgässner, 1992; Kliemt, 1986). The important distinction between OSS and other cases of collective invention consists in OSS remaining a low-cost situation for a considerable number of programmers, even when the commercial phase is reached (Lüthi, 2005), due to the three characteristics of OSS mentioned in Section 2.

The characteristics of information products keep costs of diffusion low. Participants simply post their contributions on the appropriate Internet site. Due to user innovation, the losses stemming from sharing intellectual property rights by using an OSS license are often low compared to the gains from the expected feedback by other participants. These characteristics help to keep the costs of disclosing knowledge low. Because of modularity, the costs of the production of the source code are also kept low. A modular architecture invalidates "Brooks' Law" that "adding manpower to a late software project makes it later" (Brooks, 1975). With a non-modular architecture, having more people involved in a project means higher coordination costs that can, in the extreme case, render marginal returns of manpower on productivity negative. Modularization makes useful contributions possible with reasonable integration costs.

#### 6.2.2. How the second order social dilemma is solved in open source projects: open source licenses and voluntary rule enforcing

The second order social dilemma consists of developing and maintaining the rules of voluntary cooperation. Creating and enforcing a code of ethics is itself a public good and thus constitutes a social dilemma of a higher order: "Punishment almost invariably is costly to the punisher, while the benefits from punishment are diffusely distributed over all members. It is, in fact, a public good" (Elster, 1989, p. 41).

Low-cost situations raise voluntary donations. But even in a low-cost situation, people do not contribute to public goods if no rules exist preventing them from being exploited by others. Empirical evidence shows that many individuals contribute voluntarily to public goods, as long as other individuals contribute too (Frey and Meier, 2004). They are conditional cooperators (Fischbacher et al., 2001). This cooperation, however, is rather unstable, since there is always the risk of a golden opportunity. In the collective invention context, a golden opportunity might take the form of a development of such great stand-alone value that the inventor is not willing to share it with others but prefers to appropriate its commercial value for himself. If the inventor can successfully exclude others from its use, collective invention efforts will not be sustainable.

This is impressively illustrated in the case of steam engine development (Nuvolari, 2003). Steam engine technology was already widely used and experimented with in Cornwall in the first half of the 18th century to

solve drainage problems in coal mines. These early versions of the steam engine consumed large amounts of fuel, determined by the necessity of alternately heating and cooling the cylinder. In 1769, James Watt found a solution to this problem by introducing a separate condenser. Watt patented this invention, which he and his business partner Boulton refused to license, thus frustrating the attempts of other engineers to improve on the design. Only after Watt's patent had expired could they improve on the design in a collective trial and error process, very similar to the development of iron furnaces in Cleveland. Until the middle of the 19th century, the fuel efficiency of steam engines was quadrupled. Interestingly enough, entrepreneurs at that time were unwilling to use patented technology.

How do OSS projects prevent such golden opportunities from disrupting the collective invention process? There are two factors; firstly the institutional invention of the OSS licenses, and secondly the fact that a sufficient number of intrinsically motivated contributors voluntarily enforce the rules of the OSS community.

*6.2.2.1. Open source licenses.* When people contribute their developments to OSS projects, they do not simply give up their intellectual property rights. Instead, they use their property rights to protect their developments from ever being appropriated. This is achieved by the use of OSS licenses. "Copyleft" licenses like the GPL enforce every program that contains a free software component to be released in its entirety as free software. Thus, they ensure that software derived from a public good, will remain a public good. To this day, the GPL is the most widely used license. Lerner and Tirole (2002) found that of the 25,792 open source projects they studied 70% use the GPL. While the GPL goes the longest way in protecting a project from appropriation, its viral qualities might make commercial players reluctant to use GPL'ed code. To further endorsement in the commercial world, some open source projects choose less restrictive licenses. They allow users to modify the code and sell this modified version as a proprietary product. But even the least restrictive OSS licenses ensure that what once enters the public domain cannot be re-appropriated as a private good.

In that way OSS licenses hinder exploitation of voluntary donors. They impose on the virtual community of OSS users what Hansmann (1980) calls the "non-distribution constraint", which is characteristic for non-profit organizations. The "non-distribution constraint" is a major institutional precondition for voluntary donations to organizations since it ensures conditional cooperation. It is the reason why institutions like the Red Cross or most universities are governed as non-profit organizations.

Also, for commercial providers who are dependent on the goodwill of the developers, it is crucial to make a credible commitment to the "non-distribution constraint". Thus, OSS licenses enable coexistence and fruitful cooperation between commercial providers and voluntary donators (Franck and Jungwirth, 2003).

*6.2.2.2. Voluntary rule enforcing.* Rules, which hinder exploitation of voluntary donators mean little if they are not enforced. Empirical evidence shows that the rate of contributions to public goods drops dramatically in repeated interactions if defectors are not sanctioned (Fehr and Fischbacher, 2003). In particular community managed projects (as opposed to projects sponsored or managed by firms) face this problem since there is no central authority to enforce rules. This has to be done by the community itself.

If communities want to protect their work from being exploited by hijackers, they actively engage in rule enforcement (Powell, 1992). Violation of norms may consist of improper application of the license, removal of copyright terms, or hijacking OSS by including it in proprietary software. But it is important to note that not only legal, but also informal, rules are observed and enforced. OSS licenses are not merely legal documents. Rather, they are the codified expression of a norm of reciprocity. O'Mahony (2003), who studied the means OSS communities use to protect their work, comes to the conclusion that OSS licenses are not effective because of their legal qualities, but because the community is willing to accept and ensure compliance with the norms they conceive as good behavior. The idea all these licenses have in common is that "there is no limit on what one can take from the commons, but one is expected at some time, to contribute back to the commons to the best of one's abilities" (O'Mahony, 2003, p. 13). Thus, even if a license basically allows one to sell a proprietary version of an OSS project without ever having contributed anything to the project, this doesn't mean that this is acceptable behavior. It is interesting to note that being perceived as a good "OSS citizen" is important even in projects that mainly consist of commercial firms if one wants to receive support from the community. In his study of firms operating in the environment of embedded Linux, Henkel (2004) found that the two most important reasons to reveal code are: (1) to follow the rules stipulated by the license and (2) to appear as a good player in the OSS community.

K Desktop Environment (KDE) is an impressive example on how rules of cooperation can be enforced. KDE is a Windows-like desktop for Linux and other OS

operating systems developed by the OSS community. It is based on a graphical interface toolkit called Qt. Qt was developed by Trolltech, a commercial software firm. It did not comply with the requirements of OSS (Stallman, 1999). Even though the Linux community agreed that KDE was a technically excellent product, many members refused to endorse it because they did not agree with the terms of the Qt license. These members started a parallel project called GNOME that is distributed under "copyleft". Finally, Trolltech reluctantly relicensed their product. Now Qt is available under a "copyleft" license.

While the first order social dilemma of code contribution can in principle be overcome even in the presence of purely extrinsically motivated contributors, rule enforcement can only be managed in virtual communities without the need of a central authority if there are a sufficient number of pro-socially motivated people prepared to identify and sanction rule breakers. Why do people develop a pro-social commitment to certain projects? According to Deci and Ryan (2000) three conditions are relevant: (1) autonomy, (2) feeling of competence and (3) social relatedness.

(1) Autonomy: external interventions crowd out intrinsic motivation if the individuals affected perceive them to be controlling. If that is the case, autonomy and self-esteem suffer. In contrast, intrinsic motivation is crowded in if a person's feelings of autonomy are raised (for empirical evidence, see Frey and Jegen, 2001; Frey and Osterloh, 2002). In OSS projects, autonomy is high compared to proprietary projects, which usually need deadlines and time restraints to keep them on track (Lindkvist et al., 1998). Contributors choose for themselves where and what they wish to contribute. One of the norms in OSS communities is that work cannot be enforced, neither by a central authority[12] nor by monetary incentives (Himanen, 2001). But even if a developer contributes to OSS on behalf of a company, feelings of autonomy can be stronger than when working on a proprietary project. As one programmer puts it: "It's kind of cool to bring the power to where it belongs and what's really exciting about working with the hackers is that. . .if you're [a] hacker and you leave [a firm] and you've done this work for hire, then you no longer own your product. Well, with open source, you own your stuff right?" (Contributor, GNOME a GUI Desktop Project, cited in O'Mahony, 2003).

(2) Feelings of competence grow when individuals understand what they are doing and when they feel responsible for the outcome. These feelings strengthen perceived efficacy. There is significant empirical evidence showing a positive relationship between perceived efficacy and levels of contributions to collective goods (e.g. Kollock, 1998). Feelings of competence are enhanced by feedback that is not perceived as controlling. This is the case in OSS projects because of user innovation, which implements rapid feedback cycles and enables concurrence of design and testing (Kogut and Metiu, 2001).

(3) Social relatedness is of special importance for prosocial motivation. It raises group identity, which has proven to have a strong impact on the amount of contributions to common goods (e.g. Kollock, 1998).[13] The OSS community is often described as a social movement that is guided to a large extent by a strong identification with common norms of freedom of information or of fighting against the monopoly of large companies like Microsoft (e.g. Hertel et al., 2003).

In OSS projects, intrinsic motivation is not only strengthened on the side of the sanctioner but also maintained on the side of the sanctioned. It is remarkable that most kinds of sanctions are of an informal nature, but nevertheless have a significant effect on behavior. The primary vehicles are on-line mailing lists and bulletin boards. They constitute what is called a "court of public opinion" (O'Mahony, 2003). Such an institution is well suited for maintaining intrinsic motivation. Firstly, it enhances the visibility of norms of socially appropriate behavior. There is much empirical evidence that instructions as to what is socially expected raise contributions to public goods significantly (Sally, 1995). Secondly, informal sanctions have a strong expressive function that is very suitable not to alienate people from the community, even if they are reprimanded (Ostrom, 1990).

## 7. Conclusion

Open source software production is a highly successful innovation model, which represents a special case of what Allen (1983) calls "collective invention" and Von Hippel and von Krogh (2003) call "private-collective

[12] It should be noted that governance rules in open source projects differ considerably (for an overview, see Markus et al., 2000).

[13] Even a minimal definition of groups (e.g. those who prefer Kandinsky over Klee) has been found sufficient to create a group identification (e.g. Tajfel, 1981).

model of innovation". The most important distinction is that, in contrast to the other examples of collective invention, the OSS model survives the emergence of a dominant design, and thus shows a promising way to a new innovation model, which goes beyond traditional forms of markets, hierarchies and contractual strategic alliances. This model has encouraged similar endeavors like Wikipedia[14] (an encyclopedia coauthored by thousands of volunteers) and OSCar[15]. The purpose of this paper is to inquire under what conditions this new model can emerge and be sustained.

By comparing the historical development of OSS to other established cases of collective invention, we showed that knowledge-sharing with the wider innovation system can often be observed in the wake of a radical innovation. Before a technology enters its commercially viable phase, individual losses from sharing are quite low compared to the potential collective learning benefits. Once a dominant design emerges, however, collective invention modes usually receive a severe jolt. Development efforts shift from exploration to exploitation of a technology within the limits of an established architectural design. Profit motives crowd out knowledge-sharing based on reciprocity.

In the case of OSS, this process has not taken place. OSS production is now in its commercial phase but continues to exist – at least partly – as a collective invention. We have argued that in order to explain the difference between OSS and other cases of collective invention, motivational aspects have to be taken into account. Firstly, the maintenance of the collective invention model is enabled by the characteristics of OSS that create what we have called a low-cost situation in contributing to public goods. Secondly, the institutional innovation of OSS licenses maintains voluntary donations of intrinsically motivated contributors. These licenses stipulate that what enters the public domain cannot be appropriated privately. However, these rules mean little if their terms are not enforced. Rule enforcement in itself is a public good of a higher order. Thirdly, we argued that, in order to overcome this problem, there must be a sufficient number of pro-socially motivated contributors who accept and enforce the rules of the community. The conditions of autonomy, feelings of competence and social relatedness, that are characteristics of OSS production, foster that kind of pro-social motivation.

Thus, the continuing success of the collective invention regime in OSS depends to a large extent (a) on a balance between intrinsic and extrinsic incentives to contribute to the first order public good (source code) and (b) the pro-social intrinsic motivation of a sufficient number of participants to contribute to the second order public good (enforcing the rules of cooperation). These conditions, however, are under threat, because legal protection of intellectual property rights on software is being strengthened (Benkler, 2002, 2004; Bessen and Maskin, 2000). Strict patent protection changes low-cost situations into high-cost situations. Individual OSS developers mostly make no money with their programs so that they cannot afford legal fights (Lüthi, 2005; Osterloh and Rota, 2004). So it will be important for government regulators to consider that stricter patent protection can damage the conditions for success of this promising innovation model and its applicability to other fields.

[14] www.wikipedia.org.

[15] OSCar is a project to develop an "open source car" (www.theoscarproject.org). While this project was rather active and successful for quite some time, it has been inactive since 2003. It is likely that this project was a victim of the change-over from a low-cost to a high-cost situation, once it was necessary to actually build a prototype to continue the design process.

## References


Abernathy, W.J., Clark, K.B., 1985. Innovation: mapping the winds of creative destruction. Research Policy 14, 3–22.

Abernathy, W.J., Utterback, J., 1978. Patterns of industrial innovation. Technology Review, 40–47.

Allen, R.C., 1983. Collective invention. Journal of Economic Behaviour and Organisation 4 (1), 1–24.

Amabile, T.M., 1996. Creativity in Context: Update to the Social Psychology of Creativity. Westview Press, Boulder, CO.

Anderson, P., Tushman, M.L., 1990. Technological discontinuities and dominant designs: a cyclical model of technological change. Administrative Science Quarterly 35, 604–633.

Baldwin, C.Y., Clark, K.B., 2003. Does Code Architecture Mitigate Free Riding in the Open Source Development Model? http://opensource.mit.edu/papers/baldwinclark.pdf.

Benkler, Y., 2002. Coase's penguin, or linux and the nature of the firm. The Yale Law Journal 112 (3), 369–446.

Benkler, Y., 2004. Sharing nicely: on shareable goods and the emergence of sharing as a modality of economic production. The Yale Law Journal 114 (2), 273–358.

Bessen, J., 2002. What good is free software? In: Hahn, R.W. (Ed.), Government Policy Toward Open Source Software. Brookings Institution Press, Washington, DC, pp. 12–33.

Bessen, J., 2005. Open source software: Free provision of complex public goods. http://www.researchoninnovation.org/opensrc.pdf.

Bessen, J., Maskin, E., 2000. Sequential innovation, patents and imitation. MIT Economics Department Working Paper, Cambridge, MA.

Bonaccorsi, A., Giannangeli, S., Rossi, C., 2006. Hybrid business models in the open source software industry. Management Science 52 (7), 1085–1098.

Brooks, F.P., 1975. The Mythical Man-Month. Essays on Software Engineering. Addison Wesley, Reading, MA.

Clark, K.B., 1985. The interaction of design hierarchies and market concepts in technological evolution. Research Policy 14 (5), 235–251.

Csikszentmihalyi, M., 1975. Beyond Boredom and Anxiety. Jossey-Bass, San Francisco, CA.

Deci, E.L., Ryan, R.M., 1985. Intrinsic Motivation and Self-Determination in Human Behavior. Plenum Press, New York.

Deci, E.L., Ryan, R.M., 2000. The "what" and "why" of goal pursuits: human needs and the self-determination of behavior. Psychological Inquiry 11 (4), 227–268.

Demsetz, H., 1967. Towards a theory of property rights. American Economic Review 57, 347–359.

Dess, G.G., Beard, D.W., 1984. Dimension of organizational task environments. Administrative Science Quarterly 29 (1), 52–73.

Dosi, G., Freeman, C., Nelson, R., Silverberg, G., Soete, L., 1988. Technical Change and Economic Theory. Pinter Publishers, London.

Elster, J., 1989. The Cement of Society: A Study of Social Order, New York, NY.

Ethira, J., Sendil, K., Levinthal, D., 2004. Modularity and innovation in complex systems. Management Science 50 (2), 159–173.

Faraj, S., Wasko, M.M., 2001. The web of knowledge: An investigation of knowledge exchange in networks of practice, Florida State University Working Paper, Tallahassee, FL.

Fehr, E., Fischbacher, B., 2003. The nature of human altruism. Nature 425 (23 Oct), 785–791.

Fischbacher, U., Gächter, S., Fehr, E., 2001. Are people conditionally cooperative? Evidence from public good experiments. Economic Letters 71 (3), 397–404.

Franck, E., Jungwirth, C., 2003. Reconciling investors and donators—the governance structure of open source. Journal of Management and Governance 7, 401–442.

Franke, N., von Hippel, E., 2003. Satisfying heterogeneous user needs via innovation toolkits: the case of apache security software. http://opensource.mit.edu/papers/rp-vonhippelfranke.pdf.

Frey, B.S., 1997. Not Just for the Money: An Economic Theory of Personal Motivation. Edward Elgar, Cheltenham, UK.

Frey, B.S., Jegen, R., 2001. Motivation crowding theory: a survey of empirical evidence. Journal of Economic Surveys 15 (5), 589–611.

Frey, B.S., Meier, S., 2004. Pro-social behavior in a natural setting. Journal of Economic Behavior and Organization 54, 65–88.

Frey, B.S., Osterloh, M., 2002. Successful Management by Motivation. Balancing Intrinsic and Extrinsic Incentives. Springer, Berlin, Germany.

Ghosh, R.A., Glott, R., Krieger, B., Robles, B., 2002. Floss final report—Part 4: Survey of developers. http://www.infonomics.nl/FLOS/report/FLOS_Final4.pdf.

Gosh, R., Glogg, R., Krieger B., Robles G., 2002. Free/Libre and Open Source Software: Survey and Study. International Institute of Infonomics, University of Maastricht, The Netherlands.

Hansmann, H.B., 1980. The role of nonprofit enterprise. Yale Law Journal 89, 835–901.

Harhoff, D., Henkel, J., von Hippel, E., 2003. Profiting from voluntary information spillovers: How users benefit from freely revealing their innovations. MIT Sloan School of Management Working Paper.

Hars, A., Ou, S., 2002. Working for free? Motivations for participating in open source projects. International Journal of Electronic Commerce 6 (3), 25–39.

Henderson, R.M., Clark, K.B., 1990. Architectural innovation: the reconfiguration of existing product technologies and the failure of established firms. Administrative Science Quarterly 35, 9–30.

Henkel, J., 2004. Patterns of free revealing – balancing code sharing and protection in commercial open source development. http://opensource.mit.edu/papers/henkel2.pdf.

Hertel, G., Niedner, S., Herrmann, 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.

Himanen, P., 2001. The Hacker Ethic. Random House, New York.

Kirchgässner, G., 1992. Toward a theory of low-cost-decision. European Journal of Political Economy 8 (2), 305–320.

Kliemt, H., 1986. The veil of insignificance. European Journal of Political Economy 2 (3), 333–344.

Kogut, B., Metiu, A., 2000. The emergence of E-Innovation: insights from open source software development. Reginald H. Jones Center Working Paper, Philadelphia, PA.

Kogut, B., Metiu, A., 2001. Open source software development and distributed innovation. Oxford Review of Economic Policy 17 (2), 248–264.

Kollock, P., 1998. Social dilemmas: the anatomy of cooperation. Annual Review of Sociology 24, 183–214.

Lakhani, K., Wolf, B., Bates, J., DiBona, C., 2002. The Boston Consulting Group Hacker Survey. http://www.osdn.com/bcg/BCGHACKERSURVEY.pdf, http://www.osdn.com/bcg/BCGHACKERSURVEY-0.73.pdf.

Lakhani, K., Wolf, R., 2005. Why hackers do what they do: understanding motivation and effort in free/open source software projects. In: Feller, J., Fitzgerald, B., Hissam, S., Lakhani, K. (Eds.), Perspectives on Free and Open Source Software. MIT Press, Cambridge, MA.

Langlois, R.N., 2002. Modularity in technology and Organization. Journal of Economic Behavior and Organization 49, 19–37.

Larsson, R., Bengtsson, L., Henriksson, K., Sparks, J., 1998. The interorganizational learning dilemma: collective knowledge development in strategic alliances. Organization Science 9 (3), 285–304.

Ledyard, J., 1995. Public goods: a survey of experimental research. In: Roth, A., Kagel, J. (Eds.), Handbook of Experimental Economics. Princeton University Press, Princeton, NJ.

Lerner, J., Tirole, J., 2002. Some simple economics of open source. Journal of Industrial Economics 50 (2), 197–234.

Lindenberg, S., 2001. Intrinsic motivation in a new light. Kyklos 54 (2/3), 317–343.

Lindkvist, L., Söderlund, J., Tell, F., 1998. Managing product development projects: on the significance of f'fountains and deadlines. Organization Studies 19 (6), 931–951.

Loebbecke, C., Fenema, P., Powell, P., 1998. Knowledge transfer under co-opetition. In: Larsen, T., Levine, L., De Gross, J. (Eds.), Information Systems: Current Issues and Future Changes. Laxenburg, Austria, pp. 215–229.

Lüthi, R., 2005. High Cost to Low Cost and Back? The Threat of Regulation to Low Cost Software Production. Working Paper University of Zurich, http://www.iou.unizh.ch/orga.

Markus, M.L., Manville, B., Agres, C.E., 2000. What makes a virtual organization work? Sloan Management Review 42 (1), 13–26.

Meyer, P.B., 2002. Collective Invention in History and Theory: Bessemer Steel and Open-Source Software. U.S. BLS working paper. http://econterms.com/pbmeyer/research.html.

Meyer, P.B., 2003. Episodes of Collective Invention. US BLS working paper no. 368. http://opensource.mit.edu/papers/meyer.pdf.

Moon, J.Y., Sproull, L., 2000. Essence of distributed work: the case of the Linux kernel. Firstmonday 5 (11).

Narduzzo, A., Rossi, A., 2003. Modularity in Action: GNU/Linux and Free/Open Source Software Development Model Unleashed. http://opensource.mit.edu/papers/narduzzorossi.pdf.

Nooteboom, B., 2000. Learning by interaction: absorptive capacity, cognitive distance and governance. Journal of Management and Governance 4, 69–92.
North, D.C., 1981. Structure and Change in Economic History. Norton, New York, NY.
North, D.C., 1990. Institutions, Institutional Change and Economic Performance. Cambridge University Press, New York, NY.
Nuvolari, A., 2002. Collective Invention Ancient and Modern: A Reappraisal. ECIS working paper. http://www.sussex.ac.uk/users/prff0/industrie%20studies/presentationAlessandro%202002.pdf.
Nuvolari, A., 2003. Open Source Software Development: Some Historical Perspectives. http://opensource.mit.edu/papers/nuvolari.pdf.
Olson, M., 1965. The Logic of Collective Action. Harvard University Press, Cambridge, Mass.
O'Mahony, S., 2003. Guarding the commons: how community managed software projects protect their work. Research Policy 32 (7), 1179–1198.
O'Reilly and Associates Inc., 1999. Open Source: kurz und gut, O'Reilly and Associates, Köln.
Organ, D.W., Ryan, K., 1995. A meta-analytic review of attitudinal and dispositional predictors of organizational citizenship behavior. Personnel Psychology 48 (4), 776–801.
Osterloh, M., Frey, B.S., 2000. Motivation, knowledge transfer, and organizational firms. Organization Science 11, 538–550.
Osterloh, M., Rota, S., 2004. Trust and community in open source software production. Analyse & Kritik – Zeitschrift für Sozialtheorie special issue on. Trust and Community on the Internet 26, 279–301.
Ostrom, E., 1990. Governing the Commons: The Evolution of Institutions for Collective Action. Cambridge University Press, New York, NY.
Powell, W.W., 1992. Neither market nor hierarchy: network forms of organization. Research in Organizational Behavior 12, 295–336.
Raymond, E.S., 2001. The Cathedral and the Bazaar. O'Reilly, Sebastopol, CA.
Rose-Ackerman, S., 2001. Trust, honesty and corruption: Reflection on the state-building process. European Journal of Sociology 42 (3), 27–71.
Sally, D., 1995. Conversation and cooperation in social dilemmas: a meta analysis of experiments from 1958 to 1992. Rationality & Society 7, 58–92.
Sanchez, R., Mahoney, J.T., 1996. Modularity, Flexibility, and Knowledge Management in Product and Organization Design. Strategic Management Journal 17, 63–76.
Sen, A.K., 1974. Choice orderings and morality. In: Körner, S. (Ed.), Practical Reason: Papers and Discussions. Blackwell, Oxford, UK.
Spencer, J.W., 2003. Firms' knowledge-sharing strategies in the global innovation system: empirical evidence from the flat panel display industry. Strategic Management Journal 24 (3), 217–233.
Stallman, R., 1999. The GNU operating system and the free software movement. In: Dibona, C., Ockman, S., Stone, M. (Eds.), Open Sources: Voices from the Open Source Revolution. O'Reilly, Sebastopol, CA.
Staw, B.M., 1975. Intrinsic and Extrinsic Motivation. General Learning Press, Morristown, NY.
Stringer, R., 2000. How to manage radical innovation. California Management Review 42 (4), 70–88.
Tajfel, H., 1981. Human Groups and Social Categories. Cambridge University Press, Cambridge, UK.
Teece, D., 1987. Profiting from technological innovation: implications for integration, collaboration licensing and public policy. In: Teece, D. (Ed.), Competitive Challenge. Ballinger, New York.
Torvalds, L., 1998. FM interview with Linus Torvalds: What motivates free software developers. Firstmonday 3 (3).
Tyler, T.R., Blader, S.L., 2000. Cooperation in Groups: Procedural Justice, Social Identity and Behavioral Engagement. Hove Psychology Press, Philadelphia.
Von Hippel, E., 1988. The Sources of Innovation. Oxford University Press, New York.
Von Hippel, E., von Krogh, G., 2003. Open source software and the "private-collective" innovation model: issues for organization science. Organization Science 14 (2), 209–223.
Wellman, B., Gulia, M., 1999. Virtual communities as communities. In: Smith, M.A., Kollock, P. (Eds.), Communities in Cyberspace. Routledge, New York, NY.