The Institutionalization of Open Source

Robert A. Gehring*
Poiesis and Praxis Vol. 4 (2006), pp. 54-73

Abstract

Using concepts of neoinstitutional economics, such as transaction cost economics, institutional economics, property rights theory, and information economics, the development of the Open Source movement is investigated. Following the evolution of institutions in Open Source, it is discussed what the comparative institutional advantages of this model are. The conclusion is that it is the institutional framework of Open Source, not merely the low cost of Open Source software that makes it an attractive alternative mode of organizing the production and distribution of software and software-related services. Alternative organizations will be formed and existing organizations will be transformed to take advantage of its opportunities.

Zusammenfassung

Unter Rückgriff auf Konzepte der neuen Institutionenökonomik—Transaktionskosten, Institutionen, Eigentumsrechte und Informationsökonomik—wird die Entstehung der Open-Source-Bewegung untersucht. Die Herausbildung von Institutionen innerhalb der Open-Source-Bewegung wird hinsichtlich der dadurch induzierten komparativen Vorteile diskutiert. Es wird geschlussfolgert, dass nicht singuläre Faktoren wie etwa der niedrige Preis von Open-Source-Software für die Akzeptanz dieses Modelles ausschlaggebend sind. Vielmehr ist es die Gesamtheit der institutionellen Rahmenbedingungen, die Open Source als alternatives Modell für die Produktion und Distribution von Software attraktiv machen. Es ist davon auszugehen, dass unter Verwendung dieses Modells alternative Organisationen zur Softwareproduktion entstehen werden bzw. bestehende Organisationen sich transformieren.

* R.A. Gehring, Informatik und Gesellschaft, Technische Universität Berlin, Sekr. FR 5-10, Franklinstr. 28/29, 10587, Berlin, Germany E-mail: rag[at]cs.tu-berlin.de

1 Introduction

In June 2005 one of the largest software developing corporations in the world, IBM, announced the integration of Open Source development methods into its process of software production. In an interview Doug Heintzman, IBM Software Group's VP of Strategy and Technology, remarked:

> "We basically leveraged our rather extensive experience with the open source communities and we have borrowed many of their philosophies, strategies, tools and a lot of their culture to transform IBM's internal development practices to support global component development and promote collaboration and reuse of technology." Worthington (2005)

Seen in the light of history this development looks startling, almost ironic, for only a few decades ago IBM cultivated one of the most closed attitudes of all high technology enterprises. So, why now IBM is opening up and reorganizing its business practices? What advantages holds the Open Source model for IBM (and other industry giants)? What are the economics of Open Source IBM decided to build on?

This article assumes that it is not a single one property of Open Source software such as, for example, the absence of licensing costs, that attracts industry leaders. Rather it is the institutional framework of Open Source that makes it an attractive alternative mode of organizing the production and distribution of software and software-related services.

1.1 Enter "Open Source"

Open Source software received its name in 1997, when some proponents of the non-proprietary software model—most purely expressed in the Free Software[1] concept of the Free Software Foundation (see Stallman 1999)—were looking for better ways of marketing the concept "to people who wore ties" (Perens 1999, p. 173).

With non-proprietary software, there are no exclusive intellectual property rights, which lie at the heart of the proprietary software model. Instead, users are legally entitled to modify and redistribute the software. Users are supplied with the source code of programs in order to promote the modification, reuse and redistribution of the respective software. From the point of view of conventional software marketing, Open Source software looks almost like an antithesis, breaking every rule. No surprise then that some likened it first hand to communism (cf. anon. 2003).

Nothing could be more wrong. The copyright protection for software is the *conditio sine qua non* for the existence and sustainability of the Open Source software model (Wall 1999, p 142). Without copyright anyone would be free to take the results of other people's work and sell them for his or her own profit, without compensating the developer community. Without the requirement to

[1] Within this article, Free Software and Open Source software are used as synonyms.

give back something for distributing derivative works, Open Source software probably would suffer the "tragedy of the commons" (Hardin 1968).

The Free and Open Source software movement has many roots. Some of them reach back to the early hackers[2] of the 1960s at MIT, who set out to fundamentally challenge and eventually change the centralistic and "bureaucratic world [that] was to be found at a very large company called International Business Machines-IBM" (Levy 2001, pp 41-41). The Hackers opposed this hierarchical system. According to their ethic, information should be free and authorities were to be mistrusted (Himanen 2001; Levy 2001, p 40-41). Instead of erecting arcane and holy halls for housing the computing power of "hulking giants,"[3] as was the IBM-way, hackers promoted decentralization:

> "COMPUTER POWER TO THE PEOPLE!" (Nelson 1974)

How come an industry giant like IBM embraces the hacker spirit? Surely, profit has to do with it. Capitalism is about profit and Linux now is big business, with IBM holding significant shares of the Linux server market.[4] However, profit alone cannot explain the whole picture, where at the same time hackers affirmatively support IBM's and other industry giants's fight against claims of a competitor of intellectual property infringement,[5] and where governments all over the world—Norway, Brazil, Peru, India, and many more—declare their support for the Open Source model (anon. 2005; Ashurst 2004; Noronha 2003).

On viewing the arguments pro and contra, the impression is hard to resist that Open Source is above all a matter of policy, an argument about how best to organize the future of the information society, with software being in a leverage position. Should technology development be proprietary, provided by corporations minding their own business, without interference from the state'? Or should it be directed and controlled by the public for the public has valid claims of stakes in the outcome of shaping technology as well?[6] And if, what would be the best mode of regulation'? Should the public interest be represented by governmental bodies, as in the industrial age? Or is perhaps the Open Source movement successfully demonstrating a new way of shaping technology and industry,[7] and thereby the information society, with direct public involvement?

The means by which society governs the interaction of its agents are institutions. "[T]hey structure incentives in human exchange, whether political, social, or economic." (North 1990, p. 3). The outcome of the human interaction is directed by the (relative) costs imposed on them by the institutional framework they are operating in. If Open Source was to be an alternative way of organizing software production, distribution, and use,

[2] The term *hacker,* in its original sense, sterns from *hack,* describing the process of a technological undertaking like computer programming primarily for the pleasure of itself (Levy 2001, p. 23).

[3] A nickname for the IBM 704 mainframe computer (Levy 2001, p. 19).

[4] According to the market research firm IDC, the quarterly revenue from Linux server sales now exceeds US$ 1.2 billion. With 27.7% market share, HP takes the leading position. IBM conies second with a 19.8% market share (Dunwoodie 2005).

[5] See SCO v. IBM and *SCO* v. *Novell.* Both cases are extensively documented by the Grok Law project at http://www.groklaw.net/.

[6] For a broad view see Pool (1997).

[7] For the most comprehensive treatise on the Open Source movement yet see Weber (2004).

the institutional costs of its framework would have to be competitive if compared to the traditional model.

A closer look at the evolution of institutions within the Open Source movement, using the neoinstitutional framework of (Coase 1988a; North 1981, 1990; Williamson 1985), and others,[8] will help us to find some preliminary answers to the above raised questions. And it will give meaning to the words of Doug Heintzman: "philosophies, strategies, tools and a lot of their culture".

2 The importance of institutions

Human interactions are constraint by institutions. Institutions delimit both what is permitted and what is prohibited (North 1990, p. 4), they reduce the set of choices an individual faces in a given situation (North 1981, p. 201). "[I]nstitutions reduce uncertainty by providing a structure to everyday life." (North 1990, p. 3) Thus, they make the outcome of human interaction more predictable.

In a society built around the market-based exchange of goods, services, and capital, as capitalism is (Williamson 1985), institutions together with technology (Freeman and Louçã 2002; Williamson 1985, pp 86-90) and transaction costs (Coase 1988c), determine the costs of exchange. And it is the costs of exchange which in turn determine the costs of specialization and the organizational forms of the division of work, i.e. the organization of production (Coase 1988c).

Laws, bylaws, and other formal rules penalize rule breaking, while more informal rules such as codes of conduct, customs, common practices, and ethical norms, are implicitly enforced by making rule-obeying comparatively cheaper than rule-breaking.

An individual following its goal almost always faces the choice of obeying some rules or breaking them. As postulated by the individualistic maximizing model of neoclassic economics (Homann and Suchanek 2000, p. 49), the decision is the result of a cost-benefit and risk analysis ending in opportunistic behavior wherever beneficial.

By creating disincentives for shirking and by imposing penalties on wrongdoing, incentives are directed towards cooperative behavior and the free rider problem (see Olson 1965) is dealt with. But ideology plays an important role as well (North 1981, pp. 45-48). The individual utility function cannot satisfactorily be explained without accounting for ideology. And in absence of an ideology supplementing formal rules and compliance procedures, a stable political system would be impossible (North 1981, p. 205). Institutions are no longer viable without their supporting ideology if the costs for enforcement rise above the returns from using the institutions.[9]

The institutions are not static, they evolve and change over time. They are created by human beings (North 1990, p. 205) either explicitly, such as laws, or implicitly, such as markets. The changes in technology, resource availability, or population size and structure, all may induce significant changes in

[8] For a comprehensive survey of the literature on neoinstitutional economics see Eggertsson (1990).

[9] We can take copyright and internet file sharing to illustrate what happens if ideology breaks down in face of changes in the institutional setting (see Litman 2001).

the institutional framework by changing the relative costs (prices) of the institutions. Due to the selfish behavior of rule makers (North 1990, p. 7), path dependency,[10] and significant transaction costs, inefficient institutions can persist for long times [North (1981, pp. 14, 60) and North (1990, chapters 9-11)].

2.1 Institutions and organizations

Institutions define the rules of cooperation and competition. Organizations are the players of the game, trying to exploit opportunities by either following the rules, violating them, or attempting to change them. Within an organization, transaction costs are lower compared to an alternative *modus operandi*.[11] In the long run, the interplay of institutions and organizations[12] determines the performance of the economic system (North 1990, p. 4).

Important institutions, organizations are formed around, are contracts, firms, markets, the state, constitutions, laws, property rights, standards, conventions, norms, codes of conduct and ideologies.

Important organizations are economic bodies, political bodies, social bodies and educational bodies (North 1990, p. 5). Intermediate forms do exist as well.

3 In search of institutions in the organization of software production

As of today, the organization of software production has not been broadly explored beyond a (local or global) business perspective. Clearly lacking is, save the abundant supply of treatises on intellectual property issues, an in-depth institutional analysis of this important sector of the economy, its evolution and peculiarities (McCormack 2001, p. 75). This finding is surprising, given the economic relevance of information technology in general, and the software industry in particular.[13]

With the advent of Open Source research, the research situation started to change. Since Open Source software production is not easily explained in neoclassical economic terms, research almost naturally focused on issues of institutional economics. Incentives, property rights, organization, and transaction

[10] Path dependency is the result of the establishment of inefficient technologies as industry standards, especially where strong network effects exist. Network externalities create high switching costs often avoided by the users. Thus, an inferior technology may persist for long times (see Stack and Gartland 2003).

[11] Coase (1988c) discusses how, in the presence of positive transaction costs, the firm as opposed to the market is a superior form of organizing production for realizing gains from specialization and cooperation.

[12] Since organizations constrain the behavior of their members, they are part of the institutional framework of their members, too. Firms are organizations (Coase 1988c) as well as institutions (Williamson 1985, p. 15).

[13] The world market for packaged software was estimated at US$ 169 billion in 2001 (OECD 2002, p. 5), up from about US$ 68 billion in 1994 (ibid.) and US$ 30 billion in 1987 (OECD 1996, p. 18).

costs, became the main focus of research with property rights themes drawing the largest share of attention. Meanwhile, a large body of research literature is available (Feller and Fitzgerald 2002, pp. 1-2).

The existence of two Alternative and successful models of producing and distributing software en masse now provides us with the unique opportunity to generalize "from studies of how such activities are actually carried out within different institutional frameworks" (Coase 1980, p. 211). By pursuing a comparative institution approach one can avoid ending up in a "nirvana debate" (Demsetz 1969) about economic efficiency of institutions in software production and distribution. Fortunately, technology as a variable can be ruled out because it does not differ significantly between proprietary and Open Source development. Software is still mainly hand-crafted by developers writing lines of code. Efficiency gains and losses then have to be attributed to institutional and organizational factors rather than to technological differences.

Considering proprietary software as a commodity, the most relevant institutions governing its production, distribution, and use, are:

- The state, devising and enforcing laws within its reach.
- Property rights, as devised by the state.
- Legitimate contracts, backed by the law.
- Firms, where software is produced.
- Markets, where intellectual property, software, and labor, are traded.
- Standards, defining the space for interoperability.
- Norms, defining acceptable behavior.
- The code itself, constraining the behavior of, and the user's interaction with, and through, systems.[14]

All those institutions constrain the behavior of people dealing with software in one or another context, be it using software, selling software, or developing software. In the individual cost-benefit account, the costs imposed by institutions are compared to alternative arrangements, if present, and the individual utility is optimized.

In the following sections, the fundamental institutions of the Open Source movement and their evolution are discussed. The aim is to argue that Open Source not only offers some isolated advantages such as zero licensing fees but rather a complete set of alternative institutions. It is this set of alternative institutions that, under specific circumstances, enables superior performance in software development. The differences in institutional performance lead to new organizational forms for producing software.

4 Institutions and Open Source software

The Open Source (software) movement is fundamentally about cooperation. Within the Open Source community a new way of organizing the production and distribution of information-rich goods[15], especially software, is established

[14] See Reidenberg (1998); Lessig (1999); Kesan and Shah (2002).

[15] The term 'information-rich goods' was chosen to denote products "such as songs, computer programs, novels, or scientific articles" (Besen 1987, p. 1).

[Weber (2004, pp. 224-227) and Benkler (2002), p. 371]. Understanding the relative merits of the Open Source model, its organizational impact (Weber 2004, p. 16), and its success, means to understand its institutions and organization. In the Open Source approach, traditional institutions of capitalism—property rights, contracts, norms are modified so as to minimize information and transaction costs. They are also used to maximize incentives for follow-up innovators and users instead reserving them exclusively for original innovators.

Contrary to traditional belief, where the production of complex goods has to be managed within the hierarchy of a firm, the Open Source movement proves that voluntary cooperatives among users[16] are up to the task, too:

> "It certainly should not be that these volunteers will beat the largest and best financed business enterprises in the world at their own game. And yet, this is precisely what is happening in the software industry." (Benkler 2002, p. 371)

Without comparatively efficient institutions, such endeavor would be impossible.

Following its evolutionary path, the most relevant institutions of the Open Source movement are discussed. I start with the specific ethic of hackers the development of which began as early as during the late 1950s. The following section continues with the discussion of the invention of a specific property rights regime by MIT hacker Richard Stallman in the early 1980s. The next step was the application of this property rights paradigm to software. The success of Free Software and its sibling, Open Source software, was furthered by the adoption or creation of open standards. Finally, the global availability of Open Source development tools and Internet portals established a common base for the development processes reaching beyond software.

4.1 Ethical foundations

"The Success of Open Source" (Weber 2004) did not come about accidentally but is the result of artful institution-building initiated by the software developer Richard Stallman. Stallman joined the hacker community at MIT's Artificial Intelligence Laboratory in 1971 and stayed with it until 1983, the year he left the AI lab to found the Free Software Foundation (FSF) (Levy 2001; epilogue, pp. 415-430). The reason for his decision to leave the AI lab was that he considered the hacker culture to be corrupted and victimized by the growing proprietary software movement (Levy 2001, p. 427).

The AI lab had a hacker tradition going back to the 1950s, when students and other interested parties taught themselves programming at the early computers installed in the lab. In return for its hospitality, the hackers helped pioneering the field of interactive software development and invented many of now familiar software techniques.

[16] Benkler (2002) calls it "commons-based peer production" while von Hippel (2005b) speaks of a "user-centered innovation process" (p. 2) and "user-to-user assistance" (p. 105).

More important from the perspective of institutional analysis is the ethic this group of hackers created—the *hacker ethic*. It is based on some simple principles (Levy 2001, pp. 39-49):

1. "Access to computers—and anything which might teach you something about the way the world works—should be unlimited and total. Always yield to the Hands-On Imperative!"
2. "All information should be free."
3. "Mistrust Authority-Promote Decentralization."
4. "Hackers should be judged by their hacking, not bogus criteria such as degrees, age, race, or position."
5. "You can create art and beauty on a computer."
6. "Computers can change your life for the better."

An ethic based on such principles clearly contains an alternative view on the world if compared to what Max Weber identified as "*The Protestant Ethic and the Spirit of Capitalism*".[17] The values of technological pragmatism, freedom of information, non-discrimination, peer-recognition, and "fun" instead of profit maximization, helped a community of like-minded people, the hackers, to thrive.

Without exclusive property rights there was no necessity for formal agreements such as contracts for the exchange of program code. There were no costs of monitoring people's obeyance to the clauses of contracts and there were no costs for enforcing contracts. From a transaction cost perspective, reuse of other peoples ideas, combined with the property of digitally represented expressions of ideas to be almost costless reproduced, the hacker ethic made incremental innovations in software cheap to develop and distribute.

The introduction of property rights into this community, however, turned out to be disruptive. When money as a motive became predominant (Levy 2001, p. 429) and the hacker ethic was crowded out.[18] But the hacker ethic lived forth, either personified, for example in the figure of Richard Stallman, or organized in hobbyist clubs (Saxenian 1994, p. 34).

4.2 The GPL, the "copyleft" principle, and the creation of a code commons

Richard Stallman, after the breaking-off of the hacker community, left the Artificial Intelligence laboratory at MIT to be able to adhere to the hacker ethic.[19] He intended "to write a version of the popular proprietary computer operating system called UNIX and give it away to anyone who wanted it." (Levy 2001, p. 427)

To spread the hacker ethic, to operationalize it, Stallman invented the principle of copyleft, and embedded it in what was to become the most frequently used software license of the Free and Open Source movement, the GNU General Public License (GPL). The GPL served as a blueprint for Open Source licences of all flavors. For its central role in creating the institutions of the Open Source community the GPL needs deeper investigation.

[17] Cf. Himanen (2001) for an insightful comparison of both ethics.

[18] For a discussion of the motivation crowding-out effect see Frey and Jegaen (2000) and Osterloh and Frey (2000).

[19] Sometimes ideology trumps (pure) rationality (North 1981, pp. 50-54).

Copyleft is often described as a reversal of copyright. That is not true, however. Copyleft rather is a public declaration of renunciation of exclusivity claims in one's work on condition of reciprocity. Reciprocity is reached through conditional licensing.

The GPL contains provisions covering property rights and licensing. It is based on copyright principles so that the state will enforce it if necessary, which already happened (Ebinger 2005).

At the beginning of the GPL text[20] the preamble, reverberating principles of the hacker ethic, formulates:

> "The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software—to make sure the software is free for all its users."

The basic freedom users are assured are:

1. The right to run the program code for any purpose. (GPL, sec. 0)
2. The right to copy and distribute the program code. (GPL, sec. 1)

All those rights are granted without having to negotiate first in order to get a permission from the original developer(s). Anyone may use the code, but nobody may prevent others from using it in the same ways. That is, as long as the user respects the licensing conditions.

The GPL brings the economic nature of the public good software in line with a legal regime. Since copying digital information-rich goods is for any practical purpose costless, there can't be any "over-grazing" (in the sense of Frey and Jegaen 2000). Restricting the use of software is therefore not indispensable. If what is to be gained from keeping information or information-rich goods private property is not worth the costs, it is more economic to let lose and spare unnecessary expenses.

But the GPL goes further and grants the users of copies additional freedoms:

1. The right to modify the program code, thus creating a derivative work. (GPL, sec. 2)
2. The positive right to access to the source code of each program licensed under the GPL. (GPL, sec. 3)

Once again, the rights are granted without having to negotiate first in order to get a permission from the original developer(s).

All four clauses together give the users freedom of action that they do not enjoy under current copyright laws and common proprietary licenses. In effect, the GPL clauses draw up a commons for software source code (Lessig 2001, p. 57) where code may be copied from without considerable up-front costs.

To prevent this commons from becoming stagnant, the rights are complemented by "certain responsibilities" (GPL, preamble):

- The ‘appropriation’ through proprietary licenses is excluded. (GPL, sec. 4)
- Modified program code has also to be GPL'ed, if it is distributed. (GPL, sec. 6)[21] Thus, enhancements of the original code flow back into the commons.

[20] Cf. http://www.gnu.org/licenses/gpl.html.

[21] This clause defines the much discussed virality of the GPL, i.e. that derivative code inherits the GPL with its freedoms as its license.

Instead of becoming "over-grazed," the commons is expanded by certain acts of use.

The combination of above listed rights and duties is commonly referred to as *copyleft*.

From a competition point of view, the GPL removes barriers to market entry by promoting imitation and "forward engineering" (Samuelson and Scotchmer 2002, p. 1653). With many competing suppliers for near perfect substitutable goods in the market, the market price for copies of GPL'ed programs will be the marginal price of producing the copy plus the extra charge for its distribution, i.e. close to zero in the case of digital distribution.

The consequences are twofold:

3. On the supply side, pure pre-packaged mass-market software business models are widely undermined by GPL-implied economics. Simply charging per copy will not work in the long run. Without additional service or bundling with some costlier to reproduce offers, or proprietary code, suppliers of Open Source software will hardly make any profits from selling the code. Instead service and support oriented business models will lay the foundations of success.[22]

4. On the demand side, using Open Source programs reduces up-front costs for software installations to approximately zero. Rational software users—business users and home users as well--will be attracted by the low price. And since copying is allowed, there is no scarcity in supply, once a certain piece of code has been made available. The full market demand can be met.[23]

4.3 Licensing

Licensing is a way for software suppliers to avoid the legal liabilities associated with sales contracts. According to contract law, the supplier is liable for defects in sold goods. Defects in the software, or bugs as they commonly are called, are present in all packaged software so that vendors always would be liable, especially because implied warranties cannot be disclaimed in most countries (as opposed to the US). Additionally, within sales contracts suppliers would have to specify the scope of the property rights they own and what property rights are actually transferred, which could result in limiting their own rights (OECD 1985, p. 167). Both risks are avoided by not selling but licensing software.[24]

The GPL contains a liberal framework for licensing. First of all, the GPL addresses everyone. There is no closed group, the use of the license would be reserved for. Section 5 maintains that GPL-licensing is an all-or-nothing solution with no space for negotiation. Modifying and distributing a GPL'ed program constitutes an agreement to accept the licensing terms.

If there are no negotiations, as in the case of GPL'ed code, then there are no transaction costs due to negotiating. Licensing Open Source code therefore is absolutely cheap. There are no contracts to be signed or `click-signed.' Acceptance of standard licensing clauses is sufficient.

[22] That coincides with an industry-wide trend towards computing services (see Carr 2005).

[23] The Open Source web browser Firefox impressingly demonstrates this with its fast growing installation base. See the Firefox homepage at http://www.mozilla.org/products/firefox/.

[24] IBM paved the way for licensing with their unbundling policy in 1969, cf. Grad (2002).

4.4 Property rights and the Open Source movement

Property rights are one of the foundational institutions of market-based economies (Williamson 1985). They are modeled as a bundle of rights, the exercise of which is reserved for the holder of the rights, or owner of a resource (Cooter and Ulen 2004, pp. 77-78):

- the right to use a resource,
- the right to prevent others from using the resource, and
- the right to transfer the property rights in the resource.

According to the neoclassical conception, property rights are necessary in order to enable welfare maximizing use of scarce resources by voluntary transfer of assets (Alchian and Demsetz 1973; Cooter and Ulen 2004, pp. 80-85).

For information-rich goods property rights are mainly defined by intellectual property laws granting copyrights for works of authorship to their creators and patents to inventors. As opposed to property rights in physical assets, intellectual property rights usually are not fully transferred but rather licensed. Licenses are permissions by rights-holders to use a resource in a way that normally would be unlawful (Garner 1999, p. 350).

Information-rich goods, especially information goods,[25] are public goods for they have two characteristic properties (Salvatore 2003, p. 611):

> Non-rivalry: The consumption of an information good by one individual does not interfere with the consumption of the same good by other individuals at the same time.
>
> Non-exclusion: It is difficult and expensive, sometimes prohibitively expensive, to exclude free riders from the consumption of the good.

Given both properties, the private appropriation of returns on investments in public goods may be impossible. The expectation of losses demotivates potential investors thus raising the probability of an underprovision of the public good. The result would be a market failure.

The traditional economic rationale[26] for copyright (for intellectual property in general) is to prevent this market failure. It builds on the assumption that without strong monetary incentives for original creators undersupply of knowledge products would result (Landes and Posner 2003, p. 11). Extrinsic (monetary) incentives for creators are created by granting them a term of exclusivity to market copies of their works. In exchange for publication they receive protection against imitators. That privilege permits the creators of intellectual property to charge for copies a price higher than the marginal cost of reproduction (Landes and Posner 2003, p. 11). Thus they are able to extract

[25] Information goods are information-rich goods without physical embodiment, for example music when transferred over networks.

[26] To be precise, one has to distinguish between the anglo-american copyright regime and the continental *droît d'auteur* tradition, the first one building on a utilitarian philosophy while the latter one embracing a natural law assumption.

rents from selling copies of their works, what permits to recoup the costs for producing originals.

The Open Source model on the other hand rests on the implicit assumption, that in broad areas of human creative activities, intrinsic motivation already is sufficient to "promote the progress of science and the useful arts,"[27] and no additional monetary (i.e., extrinsic) incentive is necessary in the first place. Instead intrinsically motivated people will build on what they find if permitted to do so.

By promoting an inclusive rather than an exclusive notion of property, that is by treating existing digital information-rich goods as a common pool resource,[28] property rights are devised so as to economize on existing resources. By encouraging copying, sharing, and reuse, the maximization of resource use is aimed at. As a cumulated result of small-scale, independent improvements, user-driven innovation is promoted.[29]

At the same time, the GPL takes a stand against patent protection for software (GPL, preamble).

If redistribution is not possible without patent infringement or without having to pay royalties for patents, then redistribution is not allowed (GPL, sec. 7). To avoid patent infringing distribution, copyright holders may restrict the distribution of their GPL'ed code so as to exclude certain countries.

While the patent term is much shorter than the copyright term, patents provide a much stronger protection mechanism because even independent and similar developments are covered.[30] With patents the development process would either have to include a cost-intensive patent search or an additional (expensive) step of division of labor, by outsourcing the patent search to patent lawyers. Both would add to development costs. The transaction costs of monitoring and enforcing compliance would be remarkable and perhaps prohibitive.

To sum up, the property rights model of the Open Source movement imposes lower institutional costs on its users if existing code is being reused. Copyright and copyleft licensing serve as a shield against the enclosure of the code-commons created by its contributors. The greater freedom it thereby grants the users and developers obviously fulfil their needs[31] and the duties it imposes on them makes the model sustainable.

[27] Cf. U.S. Constitution, art. 1, §8, cl. 8.

[28] For short treatises on common property see Eggertsson (1990) and Libecap (1998). For a comprehensive treatise see Ostrom (1990).

[29] Innovation in Open Source software is largely user-driven. Thereby, not only agency costs can be avoided, but an individual best-fit solution is possible, cf. Weber (2004) at pp. 265-267. When local solutions are propagated through distribution, global innovation takes place, cf., e.g., Franke and von Hippel (2003); Foray (2004) at p. 178, and von Hippel (2005a).

[30] "[C]opyright protects the expression of ideas; patent protects the ideas themselves." Stobbs (2000) at p. 28. For any given idea there are almost infinite ways of expressing it, what makes copyright protection comparatively weak. Under a patent system, however, even independent development may constitute infringement. National Commission on New Technological Uses of Copyrighted Works (1979) at pp. 16-17.

[31] A recent study by Evans Data revealed that 56.2% of software developers are using Open Source modules, up from 38.1% in 2001 (Kuchinskas 2005).

4.5 Standards

The more complex a technology is and the more dispersed the needs of its stakeholders are the more important are standards as a means of coordinating demand and supply (Foray 2004, pp. 44-45).

Standards are essentially a means of reducing transaction costs: first, by reducing the costs of measurement through signaling; second, by enabling independent development and production of components and complementary products and services (i.e. specialization and division of labor) by defining interfaces; and third, by facilitating mass production.[32] Thereby, standards contribute to preventing duplication of investment and waste of resources.

In the case of network goods such as software, standards also help creating network externalities. And, conversely, network effects help creating standards (Shapiro and Varian 1999, chapter 9, pp. 261-296). In the latter case so called *de facto* standards, as opposed to formally agreed-upon *de jure* standards, emerge, often being the result of the dominant position of a market player (OECD 1985, p. 149).

Standards may be pro-competitive or anti-competitive. Depending on their use, standards may promote or hinder the production, dissemination, and integration,[33] of complementary products for dominant platforms (David and Steinmueller 1994, p. 220). Standards can be open, i.e. free for all to adopt, or they can be proprietary, i.e. to adopt only with someone's permission.

Open standards are "a declared neutral zone…a domain in which competition through innovation will not take place" (Clapes 1993, p. 264). Existing intellectual property has to be made freely accessible if one particular technology is to become part of an open standard (Clapes 1993, pp. 264, 265). Since open standards are free for all to adopt, they are pro-competitive. Hence, open standards bring about "in most cases, more technological progress and more price competition." (OECD 1985, p. 15).

In essence, open standards aim to reduce the set of choices of suppliers in order to enlarge the set of choices for customers—both integrators and end-users as well. The counterpart of open standards are closed, proprietary standards, i.e. standards the adoption of which is not free for all competitors. Such standards usually are the result of one supplier (or a group of suppliers) holding a major market share and attempting to exclude competitors.

Developers in the Open Source community come from all over the world. They work asynchronously on different parts of complex systems and networks the integration of which requires interoperability. Therefore conformity to a common set of standards is paramount. For that reason, the Open Source community strongly promotes open standards either by adopting them or by creating them. The second important reason is the independence from suppliers of proprietary technology.

[32] See David and Steinmueller (1994) for a detailed discussion.

[33] Foray (2004), at p. 36, qualifies standards as "integrative knowledge."

4.6 Code

Code, like other institutions, defines what users and developers can do with and within a system (Lessig 1999).

The institution code takes different shapes in our context. First, there is the *source code* of software describing its functionality, i.e. the lines of code containing the instructions for the computer. In the Open Source community this code is made available for anyone to modify. Therefore, in Open Source, pure source code is not a limiting factor for individual human beings when making choices: modification of functionality is always possible. Appropriate skills are necessary, though.

Second comes the *architecture* of Open Source tools and systems. Though there are quite a lot of definitions on what the term architecture covers with regard to software, the meaning usually includes quality attributes (Albin 2003, p. 9).

Architecture is an independent property of code. Different architectures can provide the same functionality. The composition of the code, the interaction of components, and the constraints on both, are part of the architecture. For the skills of developers differ, the choices made for the architecture influence the relationships among developers (Hohmann 2003, pp. 5--6). The less an architecture requires specific knowledge the greater is the pool of potential developers. In a sense, the architecture of a system is the collection of norms governing its code, some kind of metacode.

And third, code refers to the *technical means* for actually handling the Open Source development and distribution process. It is the tools that make the Open Source model feasible. The Internet as the medium for communication, the code repositories on the Internet,[34] and the local development tools are all part of this tool box. Economically, the existence of central repositories and web portals[35] as entry points to the Open Source code base has the effect of reducing the transaction costs not only for developers but for other users too. Search costs are minimized and project properties are made transparent. As Google has changed information retrieval on the Internet, Open Source portals have changed source code retrieval.

UNIX code as paradigm

Though there are other important projects too, due to its relevance as (indirectly) having been the starting point for the Open Source movement and imprinting on it its techno-philosophical codex, the UNIX operating system deserves special attention.

When Richard Stallman decided to create a free operating system he chose UNIX as the template, as mentioned earlier. The background for his choice was a technical as well as a political one.

In the beginning UNIX was developed by three AT & T Bell Labs researchers, Ken Thompson, Dennis Ritchie and Rudd Canaday.[36] Ken Thompson implemented the prototype of what would become known as UNIX in the summer of 1969 within 1 month (Salus 1995, p. 10).

[34] The code repositories usually contain a version control system for coordinating contributions of source code changes from distributed developers. The best known of its kind is the Concurrent Versions System (CVS). See Bar and Fogel (2003).

[35] See, e.g., the SourceForge portal http://sourceforge.net/ and the FreshMeat portal http://freshmeat.net/.

[36] The history of UNIX is reconstructed by Salus (1995).

Owed to the necessity to economize on work force and computing power, the new operating system followed the approach of "trying to build neat small things instead of grandiose ones.''[37] Sometimes referred to as the KISS philosophy (for 'keep it small and simple'), this design methodology would soon prove to be very useful as a paradigm supporting the distribution of innovation.

A consent decree with the U.S. government effectively barred AT&T from selling software. Or at least that was how the AT&T lawyers interpreted the terms of the consent decree: "No business but phones and telegrams'' (Salus 1995, p. 58). So when after a presentation on UNIX Ken Thompson gave in 1973 a number of requests for licensing the operating system came in and they were answered positively but on very elemental conditions: without support and payment to be made in advance (ibid). Any support or bug fixes were denied, the users were on their own. That policy

> "had an immediate effect: it forced the users to share with one another. They shared ideas, information, programs, bug fixes and hardware fixes.'' (Salus 1995, p. 65)

UNIX quickly became a success beyond the AT&T computing department, even in Europe, Australia and Japan. Besides its technical merits, UNIX was essentially available for free. Copying fees were nominally and the system came with source code. UNIX was Free Software in the sense of Richard Stallman's philosophy.

For its users the access to the source code made it possible to port the system to other platforms. And the denial of support made it necessary to find and remove bugs on one's own.

UNIX was the perfect platform for computer science research: its architecture permitted a piecemeal development and integration process, the interfaces were documented, and its code could be studied. UNIX was an operating system that gave its users control and flexibility by design and its modular architecture was open to independent enhancements.

After the split-off of the 'Baby Bells' from the AT&T mother in 1983 the consent decree lost its base and AT&T decided to enter the hardware and software business. UNIX was commercialized. What was free before now became exclusive property. For Richard Stallman, after his experience with the destruction of the hacker community at MIT (Lohr 2001, pp. 212-213), UNIX became almost a natural target for his "war against proprietary software'' (ibid, p. 212). Stallman aptly called his project GNU, for *GNU is Not UNIX*.

Not the least important factor was that UNIX was not protected by patents that could have blocked Stallman's efforts.[38] Even if AT&T held the copyrights in UNIX, an independent implementation was perfectly legal. By personally writing the GNU operating system under a copyleft license, Stallman could remove the last barrier to what he considered the founding stone of a free software world. The GNU code would be the key to user's freedom: the users would be freed from the restrictions proprietary software imposed on them.

[37] That refers to the experiences with the development of the MULTICS operating system, cf. Salus (1995) at p. 11.

[38] First software patents were upheld in US appeals courts as early as 1976. See *In re Noll*, 545 F.2d 141 (CCPA 1976) and *In re Chatfield*, 545 F.2d 152 (CCPA 1976).

As well documented today (Moody 2001; Torvalds and Diamond 2001), the Linux operating system kernel, the development of which began in 1990 on initiative and under leadership of finish student Linus Torvalds, filled the central gap of what Stallman had started almost a decade ago. The existing parts of GNU and the Linux kernel, both using UNIX interfaces, fitted together to a complete operating system. They did not only fit in a technically but also in a legal sense because Torvalds had placed his code under the GPL thus donating it to the code commons Stallman's had projected. The copyleft principle required all other contributors to the Linux code base to do the same—a key factor in the fast growth and maturing of the Linux kernel.

Albeit only partially welcomed by the FSF protagonists, together the GNU environment and the Linux kernel fulfilled Stallman's 10-year-old plan of delivering a free UNIX to the world. For all their personal differences, both Stallman and Torvalds are working within the same institutional framework. It is a framework that differs fundamentally from the software-as-business environment where the giants of proprietary software reigned almost uncontested. The constraints imposed on the developers/users operating within the Open Source community differ significantly from their counterparts in the proprietary model. There are other choices to make, other costs to bear, and other benefits to derive.

5 Summary and outlook

It is a truism, that

> "the structure of an industry may change rapidly as costs shift.'' (Carlton and Perloff 2000, p. 6)

With the emergence of the Open Source movement and the institutional framework it created, and creates, the cost structure for software production and distribution has fundamentally changed. There is no reason then to expect the organization of software production remaining unchallenged.

As a major consequence of the establishment of the Open Source institutional system as described earlier, the aggregated transaction costs for identifying a production factor (an information artifact), measuring its performance, judging its utility, transferring the property rights for using a copy, reproducing it, and perhaps modifying it, are much lower than within the market/price system. Whenever software is neither exclusively produced nor consumed, i.e. whenever software is transformed and re-fed into the production process, the institutional framework of the Open Source movement provides a competitive option. And other areas of research and production of information-rich goods will likely profit from adopting the Open Source approach, too.[39]

With the adoption of Open Source solutions under conditions of competition with proprietary products it is now clear that the development of large, mission critical software systems does neither need the firm nor the market. By generally discouraging the use of the price-market-mechanism for factor allocation,[40] the

[39] See von Hippel (2001); Chesbrough (2003); anon. (2003).

[40] The GPL forbids to eharge license fees, cf. GPL section 11. Sec also Jaeger and Metzger (2002) at pp. 47-48.

organization of software production within classical structures, especially within traditional firms and markets, is discriminated against.[41] Instead it is possible for a network of stakeholders, individuals as well as other kinds of organizations, including firms and ephemeral ad hoc organizations, to manage an Open Source code base as a common pool resource. Within this software ecosystem (Thomas and Hunt 2004) negotiation-free and price-free reuse of code from the code pool is favored. In the long run, that reduces the costs of producing new code and maintain existing code. While exclusive property rights provided a solution to the common-property dilemma for naturally scarce resources (North 1981, p. 86), the Open Source commons provides a complementary solution for the exclusive property dilemma of artificially scarce resources, the dilemma being the trade-off of delimiting access in order to raise incentives (Landes and Posner 2003, pp. 21-24).

Regarding property rights, the GPL does not, as often misperceived, remove copyright protection. A GPL-program copy may be possessed but the program cannot be exclusively owned. The GPL and its descendants, the so-called Open Source licences, shift power and utility in favor of the possessors of copies, i.e. the users. Thereby it brings the artifact code closer to the traditional model of property, where the possessor of an information artifact rather than its creator may exert property rights, hence countering the tendency to concentrate more and more power in the hands of creators and intellectual property rights holders.

Whether the firm/market model and the Open Source community model for software production will prevail, based on present knowledge, cannot be predicted in general. Interested parties will make decisions on a case by case basis. Wherever code reuse is of little importance the Open Source model is unlikely to be preferred because bearing the costs for taking part in the community makes no sense in one-time investments. However, where open standards, reuse of existing information, and resource sharing, foster technological progress, Open-Source like production models very likely will play an important role in the future.

Alternative organizations will be formed and existing organizations will be transformed to take advantage of its opportunities. IBM, and others, already begun. Considering their activities in consortia,[42] in the ongoing outsourcing process, in partnerships with research institutions, and their commitment to Open Source activities, the growing role of networking is evident.

The traditional, self-contained enterprise is vanishing. Even former industry leaders cannot longer innovate and survive without being part of an innovation network.[43] For most of them are users of technology as well as producers, the locus of innovation tends to shift to the users (von Hippel 2005a). The traditional intellectual property framework is ill-suited to the requirements of network-based innovation and production in the information society. Where successful product development at least in part depends on the processing of tacit knowledge residing on the demand side, when there is a tight coupling of research and development, as is the case with software projects, the integration

[41] The observation of Kooths et al. (2003) is almost right in this regard. Their conclusion, however, that Open Source software development automatically has to be an inferior mode of software production and distribution ("…leads to the substantial economic and functional deficits of the open-source model") (ibid, p. 3), is grounded in overly simplistic assumptions of software economics.

[42] See, e.g., the Eclipse project, online at http://www.eclipse.org/.

[43] See Castells (2001).

of users into the production process is key to success (McCormack 2001). The institutional framework of Open Source—the set of hacker ethic, copyleft, pool of code and tools, and open standards—together with the economics of public goods provide an answer to these challenges.

References

Albin ST (2003) The art of software architecture: design methods and techniques. Wiley, Indianapolis

Alchian AA, Demsetz H (1973) The property rights paradigm. J Econ Hist 33(1):16-27

anon. (2003) Open source software: Microsoft at the power point. The Economist. http://www.economist.com/business/displayStory.cfm?story_id=2054746. 11 Sept 2003

anon. (2005) Norwegian minister: proprietary formats no longer acceptable in communication with government. Tatle. http://www.andwest.com/blojsom/blog/tatle/agenda/2005/06/27/Norwegian_Minister_Proprietary_Standards_No_Longer_Acceptable_in_Communication_with_Government.html. 27 June 2005

Ashurst M (2004) Brazil falls in love with Linux. BBC News. http://news.bbc.co.uk/l/hi/business/3445805.stm. 01 Feb 2004

Bar M, Fogel K (2003) Open Source development with CVS, 3rd edn. Paraglyph Press, Scottsdale

Benkler Y (2002) Coase's Penguin or Linux and the nature of the firm. Yale Law J 112(3):369446

Besen SM (1987) New technologies and intellectual property: an economic analysis. RAND Note, Santa Monica

Carlton DW, Perloff JF (2000) Modern industrial organization, 3rd edn. Addison-Wesley, Reading

Carr NG (2005) The end of corporate computing. MIT Sloan Manag Rev 46(3):67-73 Castells M (2001) Das Informationszeitalter I: Die Netzwerkgesellschaft. Leske + Budrich, Opladen

Chesbrough HW (2003) The era of open innovation. MIT Sloan Manag Rev 44(3):45-41

Clapes AL (1993) Softwars: the legal battles for control of the global software industry. Quorum Books, Westport

Coase RH (ed) (1988a) The firm. The market. And the law. The University of Chicago Press, Chicago, London

Coase RH (1988b) The lighthouse in economics. In: Coase RH (ed) The firm. The market. And the law, chapter 7, pages 187-213. The University of Chicago Press, Chicago, London. Reprinted from The Journal of Law and Economics 17 no. 2 (October 1974):357-376

Coase RH (1988c) The nature of the firm. In: Coase RH (ed) The firm. The market. And the law, Chapter 2, Pages 33-35. The University of Chicago Press, Chicago, London. Reprinted from Economica 4 (November 1937)

Cooter R, Ulen T (2004) Law & economics, 4th edn. Pearson Addison Wesley, Boston

David PA, Steinmueller WE (1994) Economics of compatibility standards and competition in telecommunication networks. Inf Econ Policy 6(3 4):217-241

Demsetz H (1969) Information and efficiency: another viewpoint. J Law Econ 12(1):1--22

DiBona C, Ockman S, Stone M (1999) Open Sources: voices from the Open Source revolution. O'Reilly, Sebastopol

Dunwoodie B (2005) The state of the server market. cmswire. http://www.cmswire.com/cms/industry-news/the-state-of-the-server-market-000599.php. 07 June 2005

Ebinger T (2005) Tragen die Juristen Open-Source-Software zu Grabe? In: Lutterbeck et al. (eds) Die GNU GPL vor Gericht, chapter 4. pp 249-269

Eggertsson T (1990) Economic behavior and institutions. Cambridge University Press, Cambridge, New York

Feller J, Fitzgerald B (2002) Understanding Open Source software development. AddisonWesley/Pearson, London

Foray D (2004) The economics of knowledge. The MIT Press, Cambridge, London
Franke N, von Hippel E (2003) Satisfying heterogeneous user needs via innovation toolkits: the case of Apache security software. Res Policy 32(7):1199-1215
Freeman C, Louçã F (2002) As time goes by: from the industrial revolution to the information revolution, 1st paperback edition. Oxford University Press, Oxford, New York
Frey BS, Jegen R (2000) Motivation crowding theory: a survey of empirical evidence (revised version). Institute for Empirical Research in Economics, University of Zurich, Working Paper No. 49
Garner BA (1999) A handbook of business law terms. West Group, St. Paul
Grad B (2002) A personal recollection: IBM's unbundling of software and services. IEEE Annal Hist Comput 24(1):64-7l
Hardin G (1968) The tragedy of the commons. Sci Mag 162:1243-1248
Himanen P (2001) The Hacker ethic. A radical approach to the philosophy of business. Random House, New York
von Hippel E (2001) Innovation by user communities: learning from open-source software. MIT Sloan Manag Rev 42(4):82 86
von Hippel E (2005a) Anwender-lnnovationsnetzwerke: Hersteller entbehrlich. In: Lutterbeck et al (eds), chapter 7, pp 449- 461
von Nippel E (2005b) Democratizing innovation. MIT Press, Cambridge
Hohmann L (2003) Beyond software architecture: creating and sustaining winning solutions. Addison-Wesley, Boston
Homann K, Suchanek A (2000) Ökonomik: Eine Einführung. Mohr Siebeck, Tübingen
Jaeger T, Metzger A (2002) Open Source Software: Rechtliche Rahmenbedingungen der Freien Software. C.H. Beck, München
Kesan JP, Shah RC (2002) Shaping code. http://ssrn.com/abstract_id=328920. Sept 2002
Kooths S, Langenfurth M, Kalwey N (2003) Open source-software: an economic assessment, vol 4. MICE Economic Research Studies. http://mice.uni-muenster.de/mers/. Dec 2003
Kuchinskas S (2005) Study: cost not only open source driver. Internet News. http://www.internetnews.com/stats/article.php/3520066. 14 July 2005
Landes WM, Posner RA (2003) The economic structure of intellectual property law. The Belknap Press of Harvard University Press, Cambridge
Lessig L (1999) Code and other laws of cyberspace. Basic Books, New York
Lessig L (2001) The future of ideas: the fate of the commons in a connected world. Random House, New York
Levy S (2001) Hackers. Heroes of the computer revolution. Penguin Books, New York
Libecap GD (1998) Common property. In: Newman P (ed) The new Palgrave dictionary of economics and the law (vol I-III), vol 1. MacMillan Reference Limited, London, vol I, A-D, pp 317-323
Litman J (2001) Digital copyright. Prometheus Books, Amherst
Lohr S (2001) Go to: the story of the math majors, bridge players, engineers, chess wizards, maverick scientists and iconoclasts- -the programmers who created the software revolution. Basic Books, New York
Lutterbeck B, Gehring RA, Bärwolff M (eds) (2005) Open Source Jahrbuch 2005. Lehmanns Media, Berlin
McCormack A (2001) Product-development practices that work: how internet companies build software. MIT Sloan Manag Rev 42(2):75-84
Moody G (2001) Rebel code: inside Linux and the Open Source revolution. Perseus Publishing, Cambridge
National Commission on New Technological Uses of Copyrighted Works (ed) (1979) Final report of the National Commission on New Technological Uses of Copyrighted Works (July 31, 1978). Library of Congress, Washington
Nelson TH (1974) Computer lib/Dream machines. Hugo's Book Service, Chicago. Revised and updated edition reprinted by Microsoft (1987)
Noronha F (2003) Free software carnival: Latin america takes to FLOSS in a big way. Linux Joumal. http://www.linuxjournal.com/article.php?sid=6915. 02 June 2003
North DC (1981) Structure and change in economic history. W.W. Norton, New York, London

North DC (1990) Institutions, institutional change and economic performance. Cambridge University Press, Cambridge, New York
OECD (cd) (1985) Software: an emerging industry, volume 9 of Information computer communications policy. OECD, Paris
OECD (ed) (1996) Information technology outlook 1995. Information computer communications policy. OECD, Paris
OECD (2002) OECD information technology outlook highlights. OECD Publications Service
Olson M (1965) The logic of collective action. Harvard University Press, Cambridge
Osterloh M, Frey BS (2000) Motivation, knowledge transfer, and organizational form. Org Sci 11(5):538-550
Ostrom E (1990) Governing the commons: the evolution of institutions for collective action. Cambridge University Press, New York
Perens B (1999) The open source definition. In: DiBona et al (eds), pp 171-182
Pool R (1997) Beyond engineering: how society shapes technology. Oxford University Press, New York, Oxford
Reidenberg JR (1998) Lex informatica: the formulation of information policy rules through technology. Texas Law Rev 76(3):553-593
Salus PH (1995) A quarter century of UNIX. Addison-Wesley, reprinted and corrected edition
Salvatore D (2003) Microeconomics: theory and applications, 4th edn. Oxford University Press, New York, Oxford
Samuelson P, Scotchmer S (2002) The law and economics of reverse engineering. Yale Law J 111(7):1575-1663
Saxenian A (1994) Regional advantage: culture and competition in Silican Valley and Route 128. Harvard University Press, Cambridge, London
Shapiro C, Varian HR (1999) Information rules. A strategic guide to the network economy. Harvard Business School Press, Boston
Stack M, Gartland MP (2003) Path creation, path dependency, and alternative theories of the firm. J Econ Issues (JEI) 37(2):487--494
Stallman R (1999) The GNU operating system and the free software movement. In: DiBona et al (eds), pp 53-70
Stobbs GA (2000) Software Patents, 2nd edn. Aspen Publishers, Gaithersburg, New York
Thomas D, Hunt A (2004) Open source ecosystems. IEEE Software 21(4):89-91
Torvalds L, Diamond D (2001) Just for FUN—the story of an accidental revolutionary. HarperCollins Publishers, New York
Wall L (1999) Diligence, patience, and humility. In: DiBona et al (eds) pp 127-147
Weber S (2004) The success of Open Source. Harvard University Press, Cambridge, London
Williamson OE (1985) The economic institutions of capitalism. The Free Press, New York
Worthington D (2005) IBM turns to open source development. BetaNews. http://www.beta-news.com/article/IBM_Turns_to_Open_Source_Development/1118688437/1. 13 June 2005