JENA ECONOMIC RESEARCH PAPERS

2008 – 047

Intellectual Property Rights and Ex-Post Transaction Costs: the Case of Open and Closed Source Software

by

Sebastian v. Engelhardt

www.jenecon.de

ISSN 1864-7057

The JENA ECONOMIC RESEARCH PAPERS is a joint publication of the Friedrich Schiller University and the Max Planck Institute of Economics, Jena, Germany. For editorial correspondence please contact m.pasche@wiwi.uni-jena.de.

Impressum:

Friedrich Schiller University Jena
Carl-Zeiss-Str. 3
D-07743 Jena
www.uni-jena.de

Max Planck Institute of Economics
Kahlaische Str. 10
D-07745 Jena
www.econ.mpg.de

Intellectual Property Rights and Ex-Post Transaction Costs: the Case of Open and Closed Source Software*

Sebastian v. Engelhardt**

Abstract

The economic characteristics of software and transaction costs explain, why closed source and open source software co-exist. It is about the efficient use of a non- and anti-scarce resource. But because of ex-post transaction costs that lead to information asymmetries, some property rights regarding the resource „source code“ are not exclusively separable. Thus, the first best allocation of property rights, that would yield an optimal usage of a source code, is not realizable. Or, that is to say, a first best realization of contracts is not feasible.

Hence, open and closed source software are two second best arrangements, both with specific assets and drawbacks. The principle of closed source benefits from direct (monetary) incentives and control, but has limits in its scope (size) because of transaction costs. Open source, on the one hand, benefits from its openness that creates spillovers and enables to incorporate human capital that is not acquirable for closed source firms. On the other hand, there are costs of openness, such as coordination costs (consensus finding, etc.) the danger of free riding or under provision, or forking.

JEL-classification: D23, L17, L22, O34

Keywords: open source, intellectual property rights, transaction costs, information goods, modeling property rights

*Financial support from the Klaus Tschira Foundation is gratefully acknowledged. Furthermore, I would like to thank the participants of the sixth session of the European School on New Institutional Economics (ESNIE) 2007, as well as the participants of the Annual Congress of the Society for Economic Research on Copyright Issues (SERCI) 2007 for valuable comments and suggestions on earlier versions of this paper.

**Friedrich-Schiller-University Jena, Department of Economics and Business Administration, Carl-Zeiss-Str. 3, D-07743 Jena. E-mail: Sebastian.Engelhardt@wiwi.uni-jena.de

Contents

"If the main allocative function of property rights is the internalization of beneficial and harmful effects, then the emergence of property rights can be understood best by their association with the emergence of new or different beneficial and harmful effects."
H. Demsetz, Towards a Theory of Property Rights, p 350

"The main reason why it is profitable to establish a firm would seem to be that there is a cost of using the price mechanism."
R. H. Coase, The Nature of the Firm, p 390

1 Introduction

The software sector is characterized by the co-existence of two types of production modes, based on two different concepts of ownership: closed source software (CSS)—also called 'proprietary' software—and open source software (OSS). OSS is developed by communities that include hobbyists as well as companies, and the source code—the human-readable recipe—is 'open'. This means that everybody has access, and the right to read, modify, improve, redistribute and use it. Thus, OSS appears to be a case of a private provision of a public good. Nevertheless, firms like IBM, Sun Microsystems, RedHad, etc. use OSS based business models, i.e. make money by selling products (hardware or service) that are complements to the software. Additionally, thousands of software developers contribute voluntarily, i.e. without monetary reward.

Economic research examines intrinsic and extrinsic motivations of the OSS volunteers (Lerner & Tirole 2002, Ghosh et al. 2002, Rossi 2006), the effects of OSS on competition (Casadesus-Masanell & Ghemawat 2003, Bitzer 2004), as well as open innovations (von Hippel & Von Krogh 2003, von Hippel 2005) and firm investments in OSS (e.g. Baake & Wichmann 2004, Henkel 2006, Lerner et al. 2006, Rossi & Bonaccorsi 2006). Institutional aspects—like incomplete contracting/hold-up, different types of OSS licenses, community norms, governance of OSS projects, etc.—were also brought into focus (Weber 2004, Brand & Schmid 2005, Gehring 2006, Bessen 2006, D'Antoni & Rossi 2007). This paper contributes to the institutional literature. Using a property right point of view, the rationale for open and closed source are examined. Hence, the paper also contributes to literature on possible transfers of the 'open source' paradigm (Maurer 2008, Henkel & Maurer 2007).

One can interpret the production modes of CSS and OSS as being different kinds of "institutional arrangements" (Davis & North 1971), and distinguish them by their different use of copyright law, codified in the software licenses. This different types of ownership concept leads to different allocations of intellectual property rights (IPRs) and different modes of organization. The institutional arrangements represent strategies in use of the resource software (source code respectively) and have specific assets and drawbacks regarding individual and firm level as well as social welfare.

The property rights theory mostly concentrates on negative external effects, e.g. the widely discussed tragedy of the commons—as well as the tragedy of the anti-commons (Heller 1998)—is a negative externality story, a scarce resource story. This focus seems to draw back to Demsetz' seminal article: Although he points out in the beginning, that it is about "internalization of external costs and benefits" (Demsetz 1967, p 349), he then focuses on negative externalities (Demsetz 1967, pp 350 ff.).

But the issue discussed in this paper is about a non-scarce, to some extent even an anti-scarce resource. The argument is, that property rights regarding some kind of non-rival and anti-rival applications of software (or: of the source code) are not exclusively separable, because of ex post transaction costs and the economic characteristics of software.

The paper starts in with an introduction to the analytical framework and the characteristics of software (sections 2.1 and 2.2.1). Section 2.2.2 explains why a source code is a non- and anti-scarce resource. I then show that neither non- nor anti-rivalry as such, is a reason for the dichotomy of OSS vs. CSS, as a perfect market would lead to a welfare optimal allocation of non- and anti-rival applications. The optimal allocation of property rights, optimal defined licensee agreements respectively, are derived from this (section 2.3).

Based on this, the Coasean approach of explaining non-market coordinations is used. Thus, I ask, what kind of transaction cost driven problems limit market transactions such that this can explain the co-existence of CSS and OSS. Although transaction costs limit the internalizability of some of the positive effects, this can not explain the OSS-CSS phenomena as such (section 3.1). Hence, the argument is, that because of *ex post transaction costs* some of the property rights are *not exclusively separable*, which leads to a control problem, i.e. a de facto dilution of exclusive ownership. The principle of CSS and OSS are thus interpreted as being two different kinds of solution for this. The former maximizes control and exclusive ownership while the latter minimizes control and exclusive ownership (section 3.2).

2 Intellectual Property Rights and a Non- and Anti-Scarce Resource

2.1 The Analytical Framework

This section provides some definitions used in this paper. Although this analytical framework was developed in order to analyze the resource 'source code', it can be used to analyze *any* kind of economic resource.

2.1.1 An Economic Resource

A resource can be defined as a technically meaningful set of certain elements, e.g. a source code can be defined as a technically meaningful set of code lines. Therefore, a given source code is described as a set X. As X can be split up into subsets, there is a set of all subsets $\mathcal{P}(X) = \{A \mid A \subseteq X\}$.

Let $y = f(Z, \cdot)$ denote that $Z \in \mathcal{P}(X)$ is used for an application y. The 'use' $f(Z, \cdot)$ is one of several possible forms of transformation of Z with or without the use of *other* code lines, e.g. $y = f(Z, W)$ would be an application of the combined code line sets Z and W, which can be rewritten as $y = f(V)$ with $V = \{Z \cup W\} \neq \emptyset$. However, $y = f(Z)$ is possible as well.

Because of technical reasons, applications do not exist for all Z, the trivial example is $Z = \emptyset \in \mathcal{P}(X)$. Hence, there exists a set of technically *not* meaningful subsets of X: $\mathcal{U}(X) = \{Z \in \mathcal{P}(X) \mid \nexists\, y = f(Z, \cdot)\}$. This leads to:

Definition 2.1. The set of technically meaningful subsets of X is

$$\mathcal{X}(X) = \{\mathcal{P}(X) \setminus \mathcal{U}(X)\} = \{Z \in \mathcal{P}(X) \mid \exists\, y = f(Z, \cdot)\}. \tag{1}$$

Notice, that there can be Z with multiple applications, i.e. there can exist several $Z \in \mathcal{X}$ with $\exists!\, \mathbf{y} = (y^1, \dots, y^n)$, $y^i = f(Z, \cdot)$, $n \geq 2$.

With Definition 2.1 it is possible to define for each X the corresponding set of applications, denoted by Y:

Definition 2.2. The corresponding set of applications of X is

$$Y(X) = \{y \mid y = f(Z, \cdot), Z \in \mathcal{X}(X)\}. \tag{2}$$

As mentioned above, $y = f(Z, \cdot)$ indicates, that Z might be but does not have to be combined with other code. Therefore, a more general notation is to write $y = f(Z,\cdot) \in \{f(Z), f(V)\}$ with $V = \{Z \cup W\} \neq \emptyset$ and $Z \in \mathcal{X}(X)$. This yields the following general notation of the corresponding set of applications: $Y(X) = \{f(Z), f(V)\} = Y(Z) \cup Y(V)$.

2.1.2 Definition of Scarce, Non- and Anti-Scarce

A *scarce* ressource is a ressource with *rivalry in use*. This is the case, if the use of a $Z \in \mathcal{X}$ for any application $\check{y} \in \check{Y}$ leads to *rivalry in use*.

Definition 2.3. X is called a *scarce resource* with respect to $\check{Y} \subseteq Y$, if

$$\forall \check{y} = f(Z,\cdot) \mid \Big(Y^{new}(X^{new}) \subset Y(X)\Big). \tag{3}$$

A *non-scarce* ressource is a ressource with *non-rivalry in use*, this refers to public and club/toll goods. In such a case, the use of a $Z \in \mathcal{X}$ for any application $\tilde{y} \in \tilde{Y}$ leads to *no rivalry in use*.

Definition 2.4. X is called a *non-scarce resource* with respect to $\tilde{Y} \subseteq Y$, if

$$\forall \tilde{y} = f(Z,\cdot) \mid \Big(Y^{new}(X^{new}) = Y(X)\Big). \tag{4}$$

An *anti-scarce* ressource is a ressource with *anti-rivalry in use*, i.e. the more the ressource is used, the higher is the value of the ressource, because the set of applications increases[1] due to use. Thus, a resource is anti-scarce, if the use of a $Z \in \mathcal{X}$ for any application $\hat{y} \in \hat{Y}$ leads to *anti-rivalry in use*.

Definition 2.5. X is called a *anti-scarce resource* with respect to $\hat{Y} \subseteq Y$, if

$$\forall \hat{y} = f(Z,\cdot) \mid \Big(Y^{new}(X^{new}) \supset Y(X)\Big). \tag{5}$$

2.1.3 Property Rights

The theoretical framework of this paper has to contain (intellectual) property rights. As this paper is about software, I will always *refer to IPRs only*, although in principle the notation can be applied to PRs and IPRs.

[1]Notice, that the formulation “applications” catches qualitative as well as quantitative changes of the resource.

Following Furubotn & Richter (2005), Eggertsson (1990), Hart & Moore (1990), and others, I distinguish coordination rights (usus and abusus) from residual rights (usus fructus and alienation rights). The complete set of rights is defined as $H = \{H^c \cup H^r\}$, with H^c as the set of coordination rights, and H^r as the set of residual rights. Let $h \in H$ denote one property right.

At first, coordination rights are defined: The conjunction of a coordination right $h^c \in H^c$ with a resource X (I write "$h^c : X$") leads to a distinction-criteria between the applications that are covered by the IPR and those which are not. Notice that there is no need to know the whole set of possible applications, as the distinction-criteria yields a selecting-rule r that tells whether a $y \in Y(X)$ is covered by the IPR or not: $r : y \to [0,1]\ \forall y \in Y$. This yields

$$h^c : X \longrightarrow \{y \in Y \mid r(y) = 1\}. \tag{6}$$

For example, let $\mathbf{h}_u$ denote the vector of all usus rights, $\mathbf{h}_a$ the vector of all abusus rights, and $\mathbf{h}_{u\&a}$ all usus and abusus rights. This yields e.g.

$$\mathbf{h}_u : X \longrightarrow \{y \in Y \mid y = f(Z)\} = Y(Z), \tag{7}$$

$$\mathbf{h}_{u\&a} : X \longrightarrow \{y \in Y \mid y = f(V)\} = Y(V). \tag{8}$$

Next is to define the residual rights, where I have to distinguish between usus fructus and alienation rights: Let $\mathbf{h}_f$ denote the vector of all usus fructus rights, and π the payoff gained from y, then

$$\mathbf{h}_f : X \longrightarrow \{\pi \mid \pi = f(y),\ y \in Y\}. \tag{9}$$

The right to transfer IPRs of a resource has to be represented in a slightly different way, as it is a 'right on rights'. You can find a formal notation of alienation rights in the appendix A (p 27).

2.2 The Resource Software

2.2.1 About the Economic Characteristics of Software

Software is a good with particular economic characteristics. To simplify, one can subsume the economic properties of software as follows (for more details see von Engelhardt 2006, 2008):

- Software is a *digital good* and therefore recombinant: software products are "cumulative and emergent—new digital goods that arise from merging antecedents have features absent from the original, parent digital goods" (Quah 2003, p 19). Within this text, the terms *recombinable* and *combinable* (for the term cumulative, see page 7) are defined as follows: Let $\mathcal{S}(X)$ denote the set of all permutations of X, and $S \in \mathcal{S}(X)$ denote one permutation of X. X is called *recombinable* if $\exists S \neq X$ s.t. $\exists y = f(S)$. Of course, deriving from one given source code at least one new recombinant source code just by rearranging the code lines, is more or less a theoretical eventuality only. More likely (a part of) a source code is combined with other/new code lines.

  Definition 2.6. $Z \in \mathcal{X}$ is called *combinable* with respect to W, if

$$\exists W \neq \emptyset \text{ s.t. } \exists y = f(V),\ V = \{Z \cup W\} \neq \emptyset. \tag{10}$$

- Software is *aspatial*, thus it is infinitely expansible and therefore nonrival: once software is produced, it can be reproduced without any loss of quality and the reproduction costs are virtually zero. But there are high development and pre-launch testing costs (high first copy costs). These high sunk costs combined with the low marginal costs lead to a subadditive cost function.[2]

- Software is a *network good* with direct and indirect network effects.[3] Network effects are intimately connected with *complementarity* which "means that consumers in these markets are shopping for *systems* [...] rather than individual products" (Shy 2001, p 2, emph. in original), and

[2]Economies of scope do not necessarily support subadditivity, see Baumol (1977).

[3]As software is data processing, there is an exchange of data. This exchange happens either with other software (applications and/or operating system) or hardware or both, which requires compatibility. This implies that producers have an incentive either to use the dominant standard or try to push their own standard. Consequently a coordination problem arises: which standard should be used, will it be proprietary or open? On the supply and the demand side two different forms of network effects play a role. The first is the so-called installed base effect, i.e. the utility increases with the total number of users (producers and consumers). Second, there is a personal network effect (Westarp 2003), where the adoption decision is determined less by the total numbers of users but by the adoption decisions in the personal network.

modularity (even of parts of software) plays an important role (Weber 2004, pp 172 ff; Langlois 2002, pp 22 f). A necessary condition for 'complementarity' and 'modularity' is compatibility.

- This leads to another characteristic of software, for which compatibility and combinability are necessary (but not sufficient) conditions: If X is combinable and V is compatible to X, then X can be cumulative. X is cumulative, if V itself is part of the new X^{new}.

  Definition 2.7. X is called *cumulative* with respect to W, ***if***

  $$\exists\, W \neq \emptyset \text{ s.t. } \Big(\exists\, y = f(V),\, V = \{Z \cup W\} \neq \emptyset\Big) \wedge \Big(Z, V \subseteq X^{new}\Big) \quad (11)$$

- Software is an *information good* because the source code (the system of algorithms) is information, i.e. a human-readable recipe. But software differs from other information goods, as the average consumer does not care about the information but merely about the impact (e.g. receive and send emails rather than read the source code). This explains why software is an information good that can be sold in a state users can not read the information: closed source software is typically given away only in state of (only machine-readable) binary codes and therefore the information is 'closed'.

2.2.2 Software as a Non- and Anti-Scarce Ressource

Due to the economic characteristics of software, a source code is a *non-scarce ressource*. Obviously there is no rivalry in consumption, but there is also no rivalry in production:

(i) Because of virtual zero reproduction costs, a given first copy X is a non-scarce resource for producing $n+1$ copies. X can be copied without any loss of quality, thus $y :=$ 'copying' $\mid Y^{new} = Y$.

(ii) Because of lack of physical abrasion and being combinable, a given source code X is a non-scarce resource for further software development. X can be used (even in parts) as input stock for developing first-copies of new, derived software product:[4] $y = f(V) \mid Y^{new} = Y$.

(iii) Because of software is an information good, a given source code X is a non-scarce resource for transfer of ideas and learning, thus knowledge spillover. A source code is a list of programming solutions. For a new software project, several solutions provided by existing source code might be useful and can be used in sense of a transfer of ideas and concepts even from one programming language to another. Additionally, reading a source code of interest can be beneficial for a software engineer, as one can learn from it and thereby improve programming skills. Hence, for all 'applications' y that are transfer of ideas or learning, applies $y = f(Z) \mid Y^{new} = Y$.

Thus, a source code is a non-scarce ressource. Furthermore, software is an 'at least' non-scarce ressource, as there is not only non-rivalry in use, but to some extent even anti-rivalry[5] in use, hence it is a *anti-scarce ressource*

(iv) Because of the importance of standards and network effects, software is a network good with respect to the supply side as well as to the demand side (e.g. see White et al. 2004, Kooths et al. 2003, Gröhn 1999, Gandal 1994). If X is a network good regarding a set of 'network applications' $y^{nw} \in Y$, then the following holds: $\forall y^{nw} \mid Y^{new} \supset Y$. Thus, X is an anti-scarce resource with respect to y^{nw}, see the definition 2.5, p 4.

This is true for all network goods, or network effects respectively. One intuitive example is a network of telephone wires: The more users plug in to the network, the more applications ('call person A') are possible.

(v) Because of cumulativenes, there is anti rivalry in use of a source code: If X is cumulative with respect to any $W \in \mathcal{W}$, then X is an anti-scarce resource with respect to $\mathcal{W}$.

[4]Because of this there are whole catalogs of complete elements of programs (Gröhn 1999, p 5) and a distinct programming approach—the so-called component-based software engineering—emerged. This approach also includes the re-use of software components across producers (Romberg 2003, pp 253 ff.).

[5]Compare the following also with Weber (2004, pp 153 ff.), where one can find similar thoughts, Weber uses the term 'antirivalness'.

Proof: From the definition 2.7 of cumulative (p 7) we get

$$\forall W \in \mathcal{W} \mid \left(\exists y^c = f(V),\, V = \{Z \cup W\} \neq \emptyset\right) \wedge \left(Z, V \subseteq X^{new}\right)$$
$$\Rightarrow \left((V \nsubseteq X) \wedge (V \subseteq X^{new})\right) \wedge \left((y^c \notin Y) \wedge (y^c \in Y^{new})\right)$$
$$\Rightarrow \forall y^c = f(V) \mid \left(Y^{new}(X^{new}) \supset Y(X)\right),$$

which is just equal the definition of anti-scarcity (definition 2.5, p 4).

If a software engineer further develops a module—e.g. because of own needs—others can benefit from this improvement as long as the new piece of source code is still compatible and the new 'module' is implemented in, and improves users' software systems. Thus, a source code is a potential input stock for further software development, where the new output is (potentially) another incorporable module for the software system and at the same time is again (potential) input stock that lowers further software development costs, where the result again can be implemented, and so on.

2.3 Optimal Allocation and Optimal Licenses

This section is about IPRs with respect to a non- and anti-scarce resource in a world without transaction costs. As the paper focuses on the resource software (source code), this section is about the role of copyright based licenses, because software is traditionally protected by copyright law (Graham & Somaya 2004, p 269), and the software license agreements define the transfer of the rights.

Anyhow, in this section I argue, that there exists an optimal allocation of rights regarding non- and anti-rival applications. Hence, optimal licensees can be defined:

With respect to the non-rival applications it is about a private provision of a good without rivalry in use, hence a club good story. It is well known from standard micro-economics, that a commodity without rivalry in use should be supplied, if the sum over each individual's willingness pays (WTP) at least covers the total costs (C), hence if $\sum_{j=1}^{m} WTP_j \geq C$, with m agents.

Let's assume, that there is a finite number of applications $y^i \in Y$, $i = [1 \ldots n]$, and each of the n applications is traded in one market each. It is also assumed, that

(i) the owner of the resource can price discriminate, such that each agent pays an individual price for the application i denoted by p^i_j

and

(ii) each of the n-markets is a contestable market, such that the incumbent has to choose the lowest price-vector covering the costs.[6]

This yields

$$\sum_{i=1}^{m} p^i_j = C^i, \; p^i_j \leq WTP^i_j, \qquad \forall i = [1 \ldots n] \; \forall j = [1 \ldots m] \tag{12}$$

with C^i is the portion of y^i of the (first copy) costs of Y. Obviously this is a welfare maximizing private provision of the resource.

Additionally, anti-rival applications—the rights to use the anti-rival applications respectively—can be allocated optimal as well: It is known from network theory, that ownership can internalize network effects, because if it is possible to price every single plug-in, the network effects can perfectly be internalized by dynamic pricing, i.e. by individual price discrimination that takes into account the marginal benefits of adoption (Liebowitz & Margolis 2002, 1994, Katz & Shapiro 1994). Thus, optimal internalization requires perfect price discrimination with respect to adoption, hence each adoption must be traded separately. This can be applied to *any kind* of *positive feedback* mechanism in use. Thus, regarding to the model of this section, it is again sufficient, that the source code owner can perfectly price discriminate, the result is a welfare maximizing allocation.

Hence, there are n markets where the y^is are traded, and m agents paying a price $p^i_j \geq 0$ for each $y^i \in Y$. A price $p^i_j = 0$ implies that agent j has a zero WTP for application i and hence does not buy it at all.[7] Therefore one can

[6] Of course, assuming perfect contestable markets in context of software seems to be a somehow problematic assumption. But the argument does not change, if one allows market power: In this case the monopolist or oligopolist is able to gain an extra profit, but because of perfect price discrimination, welfare is maximized also in this case. Hence the contestable-market-assumption is not a critical one for the argument.

[7] In principle, one could also allow for 'negative prices', thus $p^i_j < 0$. This would imply, that the agent gets paid for his activity causing e.g. a cumulative effect, hence being a software developer. But in order to keep the analysis simple, this is not done here.

represent the whole economy with the payment matrix P given by

$$P = \begin{pmatrix} p_1^1 & \cdots & p_1^n \\ \vdots & \ddots & \vdots \\ p_m^1 & \cdots & p_m^n \end{pmatrix},$$

with each market represented by a column. The payment structure of an agent is given by the corresponding row and can be written as $\mathbf{p}_j = (p_j^1, \dots, p_j^n)$

One can represent the complete allocation with one matrix: Let A be a $n \times m$ matrix with $a_i^j = y_i^j$. Due to (12) we have $y_i^j = 1 \Leftrightarrow p_i^j > 0$ and $y_i^j = 0$ else. For example, with $n = 5$ and $m = 4$ one possible allocation is given by

$$A = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 \end{pmatrix}.$$

Assumed, that it makes sense to write a license agreement that covers a set of applications instead of trading each application separately, then this license agreement is optimal if it transfers IPRs to agent i such that all applications are covered the agent would pay a positive price for:

$$\mathbf{h}_j : X \to \{y^i \mid p_j^i > 0\}. \tag{13}$$

And finally, the price for this license agreement is given by

$$p_j = \sum_{i=1}^{n} p_j^i = (p_j^1, \dots, p_j^n) \cdot \mathbf{1}^T, \tag{14}$$

with $\mathbf{1}^T$ is the transpose of the one-vector.

To sum up: One can define the welfare maximal allocation of non- and anti-rival applications, and a perfect market with perfect price-discrimination would lead to this allocation. Based on this, optimal defined, copyright based licensee agreements can be defined, hence an optimal allocation of IPRs is theoretically possible as well.

3 The Role of Transaction Costs

In a world with rational individuals having complete information and knowledge, there would be bargaining and trade of each $y \in Y$. Thus, transaction costs leading to incomplete information/knowledge, give reason to the fact that property rights are defined in a license agreement, and then traded. For example, if agents only know $Y_i' \subset Y$ it makes sense to trade 'rules' rather than applications. Hence, incomplete knowledge changes the standard market game into a Bayesian market game, with the agents defining, trading and pricing the rights optimally based on their subjective expectations regarding Y. Additionally, one of the basic functions of markets is to *create new knowledge* (Hayek) by rewarding innovative use of resources, and (I)PRs enable innovators to appropriate such returns.

Anyhow, the question remains, what is so unique about software, what distinguishes software from other resources, such that 'open source' establishes. Thus, what kind of characteristics of software in combination of what kind of market imperfection because of transaction costs (TC) constitutes the conditions for the co-existence of OSS and CSS?

3.1 Incomplete Information, Transaction Costs and Limits of Internalizability

Some TC, i.e. search, bargain and information costs, limit the internalizability of some of the positive external effects. One could think, that this gives reason to the fact, that OSS exists. I will show, that this is not the case, i.e. the limits of internalizability can not explain the co-existence of OSS and CSS.

Let us start with the two effects that are responsible for the anti-rivalry. As already mentioned before, network effects can be internalized via property rights. This in principle still holds in a world with TC, simply because a new network member causes a (more or less countable) increase of the value of the network only once and exactly at the moment of plug-in. Hence, as this plug-in can be connected with a right, one can price this dynamically.

But in order to internalize the positive effects caused by the the cumulative effects described in item (v) (p 8) one would have to be able to provide the engineer a benefit each time the new source code—the new module—is incorporated in someone's software system. Obviously, TC inhibit such an internalization, especially when it is about only (very) small steps in improvement and little changes, thus small step cumulative development.

In the case of knowledge spillovers described in (iii) (p 8), it is even more obvious, that neither the point of time nor the frequency of future value creation is known. The moment of 'adoption' that causes knowledge spillovers is hard to observe and moreover the value of the 'adoption' is hard to evaluate as the results are increased skills that might lead to better future performance.

The economic value of this knowledge spillovers as well as future cumulative effects are obviously hard to measure ex ante, especially in the context of innovations, because "as Arrow himself pointed out long ago, if an innovation is truly an innovation it is impossible for a finite observer to precisely forecast it." (Dosi 1996, p 84). Thus, the positive external effects of knowledge spillovers and cumulative effects can not be internalized, as they are hard to observe and measure.

This implies, that there is a lack of internalization because of lack of internalizability. This lack of internalization, thus the existence of positive *externalities* leads to too little of the relevant activity as the social benefit is greater than the private benefit. That means, if IPRs fail to some extent to internalize positive external effects with respect to source code, one can postulate a market failure to some extent. But the degree of this 'market failure' is to some extend limited: individuals are able to form subjective expectations regarding the future benefit, just like in case of the expected gains from knowledge spillovers caused by reading a reference book or textbook for example. One can argue, that a price of such a book also reflects expected future gains from knowledge spillovers.

Anyhow, even more important is the fact that—just as the book example shows—this phenomena is not unique to software markets. A lack of internalization because of a lack of internalizability *also exists in other markets*, where we can not observe such a co-existence of open and closed source.

3.2 Ex Post Transaction Costs and the Problem of Not Exclusively Separable Rights

3.2.1 The Problem

Obviously it makes no sense to claim non-enforceable IPRs, but it also makes no sense to transfer rights, that are not exclusively separable. I call a right exclusively separable, if the right holder is de facto only able to do, what is covered by the right. Let $D_i = D_i(X)$ denote the set of applications, the right holder i is *de facto* able to do. A right is a *exclusively separable right*, if $h : X \longrightarrow Y_i^h \supseteq D_i$

Examples with respect to software are the right to use a software without changing the source code, and the right to copy a software, as usually the transfer of such rights implies that the software is given away only in state of binary codes. In case of use, there is technical copy protection in addition. Hence, who ever receives such rights is—at least to some extend—de facto *not able* to do things that are not covered by the rights. Thus, IPRs with respect to software *are* exclusively separable at least if the applications covered by the right do not imply the need for access to the source code. With respect to the list in section 2.2.2 (p 7,8) this is true for the applications described in item (i) (n+1 copies) and (iv) (network effects). But what is about applications that imply access to the source code?

Before I discuss this, let me first define not exclusive separable rights:

Definition 3.1. A right is a *not exclusively separable right*, if

$$h : X \longrightarrow Y_i^h \subset D_i \Rightarrow \exists y \in D_i \mid y \notin Y_i^h \tag{15}$$

The problem of defining exclusively separable IPRs with respect to (ii) (source code as input), (iii) (knowledge spillover), and (v) (cumulativenes) is, that such applications need access to the source code. Thus, (ii), (iii), and (v) are in principle *not exclusively separable*.

Indeed, one has to recognize, that the existence of organizations like 'code-sell.com' prove, that selling source code as input is to some extend possible, although TC probably impede a lot of such market transactions: If one wants to trade rights that belong to the set of not exclusively separable rights, one has to grant access to the source code. This leads in a sense to a *club of source code users*, and and this might affect other rights, as problems with misuse can occur. The term *misuse* refers to the problem, that the source code is used in a way that is not covered by the contract.

Transaction costs can cause misuse (because control and monitoring is costly), thus may inhibit reaching the optimal club size. A club owner normally offers only the usus of the club-good, and often keeps a right of expulsion, that is to fetch back the usus from members who did not comply with the rules. But in our case there is a de facto 'dilution of the property rights' (Picot et al. 2005, p 47): Although the original owner of the source code might be still the formal owner—i.e. formally still exclusively holds abusus, usus fructus, alienation right and the right of expulsion—, in practical there is a dilution of the IPRs, because of TC: An increase of club members induce control costs, thus with a huge amount of members it is simply not possible anymore to control if some members re-use the source code for own purpose and/or re-sell the code. The latter gives reason to the fact, that a right to expulse is not enforceable in groups with (too) many members. Hence, with an *increase* of 'cooperation-club' members c. p. the feasibility of misuse *increases* as policing and enforcement costs *increase*. Figure 1 depicts this problem:

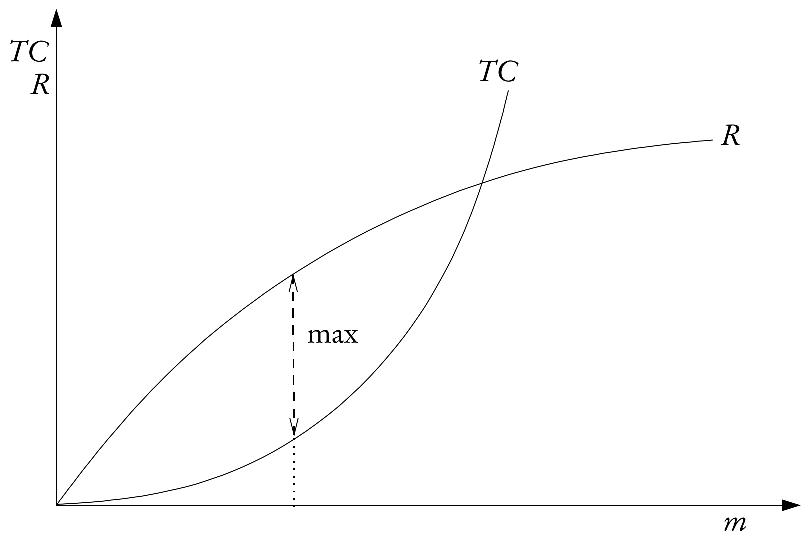

Figure 1: Transaction Costs and Limits of Size (Members)

With increasing number of members, TC increase. Assumed that the owner of the resource would first trade with the agent offering the highest price, the m agents are arranged by the price they are willing to pay. Hence, the returns as a function of club members $R = f(m)$ is concave.[8] The optimal number of members is indicated by the dotted line. It is easy to see, that if TC are high, the optimal number is small, maybe zero.

Policing and enforcement costs c. p. *decrease* with the *decrease* of specificity of the cooperation contract,[9] but a *decrease* of specificity c. p. *increases* feasibility of misuse. The first part of this logic is shown in figure 2: the TC

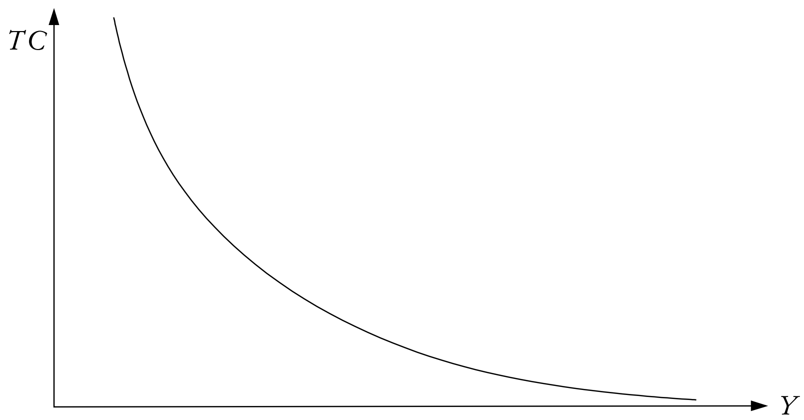

Figure 2: Transaction Costs and Specification of Contract

decrease with decrease of specificity i.e. the increase of set of allowed applications, simply because if one allows every possible application one does not have to control anything. (Notice, that for the argument it is irrelevant whether one assumes that the decreasing transaction cost curve is convex or linear.) The question is, whether a market for such unspecific licenses would establish, would be stable respectively.

The price one can get from selling somebody a licensee agreement would be maximal if this licensee would allow every possible application. Of course,

[8] The argument also holds, if one assumes, that all agents have the same WTP, hence if $R = f(m)$ is linear.

[9] This is also true for ex ante TC, namely bargaining and decision costs.

a *decrease* of specificity c. p. *increases* feasibility of misuse. But the set of applications an agents de facto can use, and hence the price an agent will be willing to pay for an complete unspecific contract, is determined by the agent's set of realizeable applications (Y_i^{tc}). This implies, if the resource owner would *know* each Y_i^{tc}, optimal PR-allocation would be reached again.

Obviously the problem is that the agents do not know other agent's Y_i^{tc}. Hence, they do not know the *type* of the other agents, and this leads to a problem well known as 'adverse selection'—because Y_i^{tc} determines ex post (hidden) action of agent i, the ex post problem can be transferred to an ex ante information problem. As there are incentives to indicate a smaller Y_i^{tc} than the real one, because this would lead to a smaller price, the agents will not truly indicate their Y_i^{tc}. The owner of the source code might know—or has a sufficiently correct idea about—the distribution of Y_i^{tc}, the average set of realizeable and tradeable applications $\bar{Y}_i^{tc}$ respectively. But given the corresponding average price, at least some agents with $Y_i^{tc} < \bar{Y}_i^{tc}$ won't pay for this, and leave the market in a sense. This increases the average set of applications and therefore the average price, and so on. This yields the well-known result of adverse selection: At the end, only the agent with the largest set of applications will rest in the market.

To sum up: Because of ex post TC, IPRs that imply access to the source code are de facto not exclusively separable. This inhibit optimal allocation of the IPRs, as the selling of not exclusively separable is limited. If one wants to transfer such not exclusively separable rights to many agents, one has to forgo the exclusivity of the rights.

3.2.2 Implications for Licensing: The Case of CSS vs. OSS Licenses

The existing software licenses can be classified by the scope of transferred rights (for the following see Hawkins 2004 p 107, Böhnlein 2003, p 19 ff, and Nüttgens & Tesei 2000, p 11):

- CSS licenses are *exclusive* as they are based on the principle of closeness, i.e. a user (licensee) of CSS typically receives only the usus and (maybe restricted) usus fructus from the licenser, and the alienation right is not transferred or restricted (see table 1).

- OSS licenses are *inclusive* as they are based on the principle of openness: The licenser offers the license to anybody who wants it, and OSS licenses *in principle* transfer the whole set of rights (see table 1). But they differ in the scope of transferred rights. *Public* OSS licenses—like the BSD license—do not restrict the use of the software and the source code in any way. So-called *viral* licenses—like the GPL—differ in the alienation rights, as the right to redistribute is restricted: Any further developed software as well any derived work must be licensed as a whole under the same type of license. Hence, OSS is not software without any property, as e.g. the GPL is based on copyright.[10] An OSS license is a contract that offers everybody the whole set of rights while the possible constraint, thus possible limitation of the alienation right, must be considered only at the moment of redistribution.

	usus	usus fructus	abusus	alienation right
CSS	+	+	-	(-)
OSS (viral license)	+	+	+	(+)
OSS (public license)	+	+	+	+

Table 1: The Transfer of Usus, Usus Fructus, Abusus and Alienation Right

The different kinds of licenses can be explained by the theory of not exclusively separable rights: CSS licenses, based on the principle of exclusive ownership, trade only such IPRs, that *are* exclusively separable, and do not allow access to the source code. OSS licenses in contrast offer the complete set of rights, as they allow access to the source code. Both type of license imply a certain kind of organization of production, thus governance structure.

The question remains, whether it can be rational to choose an OSS license, thus to forgo the exclusivity of some rights. Therefore the next section discuss the rationality to forgo the exclusivity of (not exclusively separable) rights of an non- and anti-scarce resource.

[10]The "GPL contains provisions covering property rights (...) [,] is based on copyright principles (...) [and] does not (...) remove copyright protection" (Gehring 2006, pp 62, 70). Thus, it is somehow misleading, that the counterpiece of OSS *only* is called proprietary, as 'proprietary' comes from the latin terms proprietarius and proprietas, meaning 'protected by copyrights'. Therefore I prefer to use the term closed source software.

3.2.3 On the Rationality Not to Claim all Rights

It might not cause real costs for the source code owner to freely transfer some rights. The *real* costs are only the lost revenues regarding the set of rights one could have sold. Let $Y_i^{tc} \subset Y$ denote the set of applications an agent i can realize and/or trade, can trade the corresponding IPRs respectively. The superscript tc indicates, that this set is determined by TC while the subscript i indicates that this set also depends on individual 'factors', e.g. the access to necessary resources, etc. To give an example: a student's Y_i^{tc} of a certain source code might be smaller than Microsoft's Y_i^{tc} of the same source code.

First, the role of IPRs regarding non-scarce resources is defined: Assuming that envy and (irrational) stinginess does not play a role, there is no reason why an agent i should claim and/or not freely transfer the rights $\mathbf{h}$ that cause no real costs: $\mathbf{h} : X \longrightarrow \left\{ \tilde{y} \notin \tilde{Y}_i^{tc} \mid (\nexists y^{tc} \in Y_i^{tc} \mid \varepsilon \leq 0) \right\}$ with ε as the cross-price elasticity of $y^{tc} \in Y_i^{tc}$ regarding $\tilde{y}$.

This leads to the definition of the *optimal* exclusive IPRs with respect to a non-scarce resource.

Definition 3.2. In case of a non-scarce resource, the rights $\mathbf{h}$ that should be exclusively claimed are given by

$$\mathbf{h} : X \longrightarrow \tilde{Y}_i^{\mathbf{h}} = \left\{ \{ \tilde{y}_i^{tc} \in \tilde{Y}_i^{tc} \} \cup \{ \tilde{y} \notin \tilde{Y}_i^{tc} \mid (\exists y_i^{tc} \in Y_i^{tc} \mid \varepsilon > 0) \} \right\},$$

simply because $\pi_i(\tilde{Y}_i^{\mathbf{h}}) = \pi_i(\tilde{Y}_i^{tc})$.

Thus, exclusive IPRs regarding a non-scarce resource are defined in an optimal way, if they protect (a) all applications agent i can *realize and/or trade*, and (b) all applications that are *substitutes* to any y_i^{tc}

Regarding the feedback-effects, the argument is basically the same. Additionally one has to take into account, that it can be rational not to claim exclusive IPRs regarding some $\{ \hat{y} \notin \hat{Y}_i^{tc} \mid (\exists y_i^{tc} \in Y_i^{tc} \mid \varepsilon > 0) \}$ and even regarding some $\hat{y}_i^{tc} \in Y_i^{tc}$, if the benefits (i.e. payoff) from the feedback effects (over)compensate the costs. Because of the definition 2.5 (anti-scarcity, p 4) we know that $\forall \hat{y} \mid \left(Y^{new} \supset Y \right), \Rightarrow \exists Y^{+} = \{ Y^{new} \setminus Y \}, \Rightarrow \hat{y} \rightarrow Y^{+}$.

This leads to the definition of the *optimal* exclusive IPRs with respect to a anti-scarce resource, rights covering anti-scarce applications respectively.

Definition 3.3. In case of an anti-scarce resource, the rights $\mathbf{h}$ that should be exclusively claimed are given by

$$\mathbf{h}: X \longrightarrow \hat{Y}_i^{\mathbf{h}} = \Big\{\{\hat{Y}_i^{tc} \cup \{\hat{y} \notin \hat{Y}_i^{tc} \mid (\exists y_i^{tc} \mid \varepsilon > 0)\}\} \setminus \{\hat{y} \mid \pi_i(Y^+) \geq \pi_i(\hat{y}), \hat{y} \rightarrow Y^+\}\Big\}.$$

Notice, that in order to be able to benefit from the feedback effects, one has to keep the usus fructus right.

Anyhow, taking together the facts, that some rights are not exclusively separable and that it can be rationale not to claim all rights of a non- and anti-scarce resource, then neither CSS nor OSS licenses seem to be irrational: The above definition of optimal defined property rights with respect to the non- and anti-scarce applications needs only the modification to take into account that some rights have to stay bundled. Based on this, it can be rational either to keep this bundle of rights (taking into account that the not exclusively separable rights can not be sold), or to forgo the exclusive claim, i.e. choose OSS. It is the decision between two possibilities, two second best solutions, as the first-best allocation (see section 2.3) is not realizeable.

Thus, OSS is a rational choice, if the lost of exclusive rights either cause no real costs, or the (expected) benefits[11] caused by the ant-rival applications overcompensate the real costs. Such benefits (payoffs) are the more likely,

- the smaller the set $\hat{Y}_i^{tc}$, and the more motives like reputation (Lerner & Tirole 2002), hobby reasons etc. play a role.
- the more agents cause cumulative effects because of their use of the code. Thus, the more unspecific the license agreement is.
- the more it is possible to gain profits from selling goods that are complementary to the software.
- the less the other agents are direct competitors, thus the less likely $\exists y_i^{tc} \in Y_i^{tc} \mid \varepsilon > 0$.

[11] I am not going to discuss here, how, why and when 'coordination' (thus contribution) is a equilibrium strategy in such OSS-games (that are somehow public good games). For the purpose of this paper it is sufficient to recognize, that OSS exists in the real world.

3.3 Two Solutions: A Comparison of OSS and CSS

In the last section, I explained, why individuals can not reach the first best allocation of IPRs because of ex post transaction costs that lead to the problem of not exclusively separable rights. I then showed that OSS and CSS licensees are two pragmatic real life solutions for this problem. Finally, I explained the individual rationality, i.e. the conditions for not to claim all rights, and, based on this, the individual rationality for OSS or CSS.

In this section, I also take into account welfare aspects. Based on the analysis above, I discuss the different institutional arrangements based on the OSS and CSS principles with respect to their assets and drawbacks.

3.3.1 The Principle of CSS

In case of CSS, only the exclusively separable rights are trade. Thus, these rights are sold in the market, and hence the effects of the applications covered by this rights are internalized (at least in principle).

Obviously, the resource source code is used by more than one individual having access to the source code. Here, the solution to solve the 'dilution-of-control'-problem is to increase control by adding a second set of rules, thus to build a governance structure that ensures control, i.e. build a 'firm'. Thus, CSS is based on the principle to maximize exclusive rights. Using the governance structure of a firm, it is possible to produce software as a coordinated work of several software developers, who are employees. This enables to benefit from direct control, thus the used input resources—namely human capital—are employed in a efficient way.

Of course, there are some hierarchy costs, e.g. induced by principal-agent problems. As developing software is sometimes like finding solutions for a problem, one may not be able to conclude directly from the output the effort of the developer. Software is therefore developed in a principal-agent-structure, as the principal defines certain aims and arranges a team of programmers, testers, etc., but can barely monitor the effort and/or performance of these agents, because of monitoring costs (Pasche & von Engelhardt 2004, p 9). Such hierarchy costs limit the size of a firm. Hence, a firm can not include everybody who would have been able to submit something. Obviously it is not possible to write an employment contract with everybody who would have been able to do some cumulative activities. Thus, some human capital is not acquirable for CSS firms, as the search, bargaining and enforcement costs are too high. Other kind of contract based relationships are also limited (see p 16) Thus, with respect to the maximal possible number of agents who *could generate positive effects*, a CSS firm can not reach the optimal size.

Additionally, with respect to applications covered by the not exclusively separable rights, the firm has some limits. Of course, within a CSS firm, positive effects are internalized: The firm owner not only owns X but also exclusively owns X^{new}, as the employment contracts contain a paragraph that makes sure, that all the possible copyrights are transferred to the company, thus at the end of the day they do not own any IPRs concerning the source code. And additionally to the cumulativenes aspects, the benefits of knowledge spillovers are also internalized, at least as long the employee work for the firm.

But the set of beneficiaries of the positive effects that imply access to the source code is suboptimal small. As CSS is sold only in state of binary code, no one outside the firm gets access to the source code. This is different to how copyright protection (and IPR in general) works in other fields, that is to combine ex ante incentive to produce with the ex post disclosure of the information (Cowan & Harison 2001, Quah 2003, pp 16 f, 19 ff).[12] In case of software this is not possible, because of the problem of exclusively not separable rights.

[12] Intellectual property law is defined in a way, that the rights protect the tradeable, hence private internalizeable effects, and forgoes the right for such positive effects, that are not internalizeable. Patents do not protect the idea itself but its application in form of machine, method or matter (Besen & Raskind 1991, p 12). The right to be a temporary monopolist regarding the economic use of a novel technical solution is bundled with the constraint to disclosure the information that stands behind the innovation, as the technological solution has to be described in the patent specification. Similarly, copyright does not protect the idea itself—the pure information—but its expression. Thus, in the case of e.g. a copyright protected book the author earns money from its publication, which is a disclosure of the ideas (or: information). With increasing sales figures, the author earns more money and the ideas of—the information within—that book diffuse, because everybody who buys that book can read it.

3.3.2 The Principle of OSS

OSS licenses are very non-specific cooperation contracts, designed in order to attract a large number of club members, and realize benefits from knowledge spillover and the cumulative feedback mechanism. As OSS licenses are not limited in time, they seem to be designed for a cooperation with—at least potentially—infinite duration: At any time, anyone can access the source code and use it however long, given that the possible constraint, that is a possible limitation of the alienation right, is considered.

This implies, that the principle of OSS maximizes the number of individuals, who can cause positive effects. And, with respect to the cumulative effects, the institutions of OSS support, and to some degree even guarantee it: Although one agent might be able to make money selling further developed source (hence, it causes significant real costs to give it away for free), licensees like the GPL force this agent to contribute his work back. In addition to this copyright-based institution, informal rules like the hacker ethic and community norms (incl. the enforcement characteristics) also support this cumulative effects, as at least some kind of contribution may be expected by the community. Thus, social norms can play a role here, as breaking the rules will be sanctioned by the community, that is stop cooperating or migrate to other projects (Osterloh et al. 2001, p 16 f).

Of course, as such effects are not (sufficiently) internalized, individuals involved tend to underprovide, although the number of participants is somehow maximized.

Anyhow, as OSS licenses are inclusive, with respect to the source code, the number of (potential) beneficiaries is also maximized (Of course, restricted licenses like the GPL somehow limit this, as CSS producer can not use GPL protected code as direct input).

A second implication of the inclusive OSS licenses is, that it is not possible to hold exclusive rights, and therefore it is not possible to divide neither consumers from producers, nor software developers from coordinators or dividing coordination and ownership. Everybody involved holds the complete set of rights regarding the source code.[13] This raises the question of control

[13]In case of viral licensees there is of course a limitation of the alienation right, but this limitation affects everybody, thus all involved holds the same amount of limited alienation right.

and coordination. Of course, the different OSS projects are clear structured, the basic organizational structure of such OSS-Projects is often labeled the 'onion layer' model (for details see e.g. Jensen & Scacchi 2007, Crowston et al. 2006, Wendel de Joode et al. 2003, pp 18,19). It exist clear rules about how one can move from the outermost layer into the core of the project, e.g. one has to prove software development skills, reliability etc. At the core, there exist so-called *core developers*, who oversee the design and evolution of the project. And, maybe most important, the core developers control—thus, manage—the project by using *passive control rights*, that are their exclusive rights to decide whether to accept or reject contributions (McGowan 2001, Wendel de Joode et al. 2003, p 20). The passive control rights are enforced by using the concept of ownership regarding the database in which the software is stored and the name—thus, the trademark[14]—of the project. This prevents cloning of projects and supports the signalling function of the project's name, thus trademark.

Although such passive control rights and other aspects of the governance of OSS projects (e.g. see de Laat (2007) on this topic) exist, one can state the following: Compared to CSS firms, OSS projects are likely to have higher coordination costs. The concept of openness and the inclusive licenses can lead to coordination problems like forking (a project splits up into several incompatible projects because of different goals of the individuals involved) or failed establishing of new standards. Additionally, as consensus plays an important role in OSS projects (Brand & Schmid 2005), there is the danger of 'never ending' discussions and other costs of consensus finding.

3.3.3 The Co-Existence of OSS and CSS

OSS and CSS are somehow complementary in their assets and drawbacks. OSS has weaknesses where CSS has strengths, and vice versa. This is true on an individual level as well as on social level (welfare aspects).

Thus, the co-existence of OSS and CSS can be explained by the fact, that different individuals with different sets of tradeable rights, different resources etc., need different 'solutions'. Additionally, an OSS-CSS mix can be optimal

[14] E.g. Apache is a trademark of The Apache Software Foundation, KDE and K Desktop Environment are trademarks of KDE e.V., Linux is a registered trademark of Linus Torvalds, and so on (see www.apache.org, www.kde.org, www.linuxmark.org).

from a social point of view: Whereas CSS is better in using the acquired resources efficiently namely via direct control, internalizing the positive effects, create user-friendly innovations (plug and play, easy installation routines and 'nice' graphical user interfaces) and radical innovations (because the positive effects of a paradigm change can be internalized, which enables to bear the costs of it) etc., OSS can integrate human capital CSS can not acquire, create spillovers more individuals can benefit from, and is better in more incremental technical innovations and user innovations (von Hippel & Von Krogh 2003, von Hippel 2005) etc. It is possible, that these effects are really complementary to each other, thus that a co-existence of OSS and CSS is welfare optimal. Of course, the intersting question is, whether, or under which conditions, the 'optimal' OSS-CSS mix establishes, or not. Although an intersting topic, this is beyond the scope of this paper.

4 Summary and Outlook

The paper at hand examines the rationale for open and closed source from property right point of view. Software is somehow seen as an example, where open and closed source co-exist. The findings of this paper can be summarized as follows:

1.) Because of ex-post TC and the economic characteristics of software, some IPRs are not exclusively separable. Namely IPRs covering applications that imply access to the source code can't be unbundled. Thus, the first best allocation of IPRs, that would yield an optimal usage of a source code, is not realizable. Or, that is to say, a first best realization of contracts is not feasible.

2.) Because of TC, the set of individuals one can contract with without having a de facto dilution of ownership, the 'club of source code users', is limited. Unspecific contracts concerning the code have lower TC, hence can reach more individuals. In other words: minimized exclusive ownership and restrictions enables to contract with more individuals.

3.) It can be rational, to forgo some rights, either because this does not cause any real costs, or because feedback mechanisms overcompensate real costs, or because of both. The benefits (payoff) caused by feedback mechanisms are c.p. higher, the more profits can be made by selling goods and services, that are complementary to the software. Of course, this is true for not exclusively separable rights as well.

4.) CSS trades the exclusive separable rights only. Production takes plays in firms, based on the principle of exclusive ownership of the source code. The other solution, OSS, minimizes the restrictions regarding the source code, thus OSS licenses are unspecific contracts, transferring not exclusive separable rights.

    Both principles can be a rational choice. The decision depends on the individual's set of tradeable applications/rights, the (expected) feedback effects, the individual resources, the possibility to gain profits from complementary products, etc.

5.) From a social point of view (welfare point of view), OSS and CSS are two second best arrangements, both with specific assets and drawbacks. The principle of CSS benefits from direct (monetary) incentives and control, but has limits in its scope (size) because of TC. OSS, on the one hand benefits from its openness, that creates spillovers and enables to incorporate human capital that is not acquirable for CSS firms. On the other hand, there are costs of openness, such as coordination costs (consensus finding, etc.), the danger of free riding or under provision, or forking. Hence, as OSS and CSS have their specific assets and drawbacks, an OSS-CSS co-existence can be welfare optimal.

The analysis presented in this paper also yields some further research questions: Under which conditions is an OSS-CSS mix welfare optimal? Will such an optimal mix establish, and/or will it be stable? OSS-projects are based on the principle of minimized exclusive ownership combined with passive control rights. Ho does this kind of passive control work? And finally: As more and more firms use OSS, how can such a OSS vs. CSS license decision of a firm be modeled in a simple way?

A Appendix: On the Notation of Alienation Rights

Let h^h denote the alienation right regarding h. An individual holding an alienation right with respect to h can therefore *decide* wether to keep on holding this right, or transfer it, i.e. do not hold it anymore:

$$h^h = \text{'decision'} \circ h = \begin{cases} 0 & \text{if right } h \text{ is transferred,} \\ h & \text{if right } h \text{ is not transferred.} \end{cases} \tag{16}$$

Let $\mathbf{h}^h$ be the vector of all alienation rights. Holding alienation rights on a resource X is then formally given by

$$\mathbf{h}^h : X = \begin{pmatrix} h^{h^1} : X \\ \vdots \\ h^{h^n} : X \end{pmatrix} = \begin{pmatrix} [0,1] \cdot h^1 : X \\ \vdots \\ [0,1] \cdot h^n : X \end{pmatrix}. \tag{17}$$

Notice, that there exist "recursive" alienation rights. A recursive alienation right is the right to transfer an alienation right. This means, that

$$\exists\, h^k \in \{h^1 \dots h^n\} \text{ s.t. } \left[h^k = h^{h^j} \text{ with } h^j \in \{h^1 \dots h^n\},\, h^j \notin \mathbf{h}^h \right]. \tag{18}$$

References

Baake, P. & Wichmann, T. (2004), Open source software, competition and potential entry., Berlecon Research Papers 5, Berlecon Research, Berlin.

Baumol, W. J. (1977), 'On the proper cost tests for natural monopoly in a multiproduct industry', *American Economic Review* **67**(5), 809–22.

Besen, S. M. & Raskind, L. J. (1991), 'An introduction to the law and economics of intellectual property', *Journal Of Economic Perspectives* (1), 3–27.

Bessen, J. (2006), Open source software: Free provision of complex public goods, *in* J. Bitzer & P. Schröder, eds, 'The Economics Of Open Source Software Development', Elsevier, pp. 57–81.

Bitzer, J. (2004), 'Commercial versus open source software: The role of product heterogeneity in competition', *Economic Systems* **28**(4), 369–381.

Böhnlein, I. (2003), Anwendung von Aspekten der Neuen Institutionenökonomik auf Open Source Software. Produktion, Verfügungsrechte und Transaktionskosten – eine theoretische und empirische Untersuchung, Master's thesis, Johann Wolfgang Goethe-Universität, Frankfurt am Main.

Brand, A. & Schmid, A. (2005), Koordination in einem Open Source-Projekt, Technical report.

Casadesus-Masanell, R. & Ghemawat, P. (2003), Dynamic mixed duopoly. a model motivated by linux vs. windows, IESE Research Papers 519, IESE Business School, Barcelona.

Coase, R. H. (1937), 'The nature of the firm', *Economica* **4**(16), 386–405.

Cowan, R. & Harison, E. (2001), Protecting the digital endeavour. prospects for intellectual property rights in the information society, Research Memoranda 28, MERIT, Maastricht Economic Research Institute on Innovation and Technology, Maastricht.

Crowston, K., Wei, K., Li, Q. & Howison, J. (2006), Core and periphery in free/libre and open source software team communications, System Sciences: HICCS (Hawaii International Conference) Proceedings, pp. 118a–118a.

D'Antoni, M. & Rossi, M. A. (2007), Copyright vs. copyleft licencing and software development, Working Paper of the Department of Economics University of Siena 510, Department of Economics, University of Siena.

Davis, L. E. & North, D. C. (1971), *Institutional Change and American Economic Growth*, University Press.

de Laat, P. (2007), 'Governance of open source software: state of the art', *Journal of Management and Governance* **11**(2), 165–177. Special Issue on the Governance of OSS.

Demsetz, H. (1967), 'Towards a theory of property rights', *American Economic Review* (2), 347–359.

Dosi, G. (1996), The contribution of economic theory to the understanding of a knowledge based economy, *in* OECD, ed., 'Employment And Growth in The Knowledge-Based Economy', Paris, pp. 81–92.

Eggertsson, T. (1990), *Economic Behavior And Institutions*, Cambridge University Press.

Furubotn, E. G. & Richter, R. (2005), *Institutions And Economic Theory : The Contribution Of The New Institutional Economics*, Univ. Of Michigan Press.

Gallini, N. T. (2002), 'The economics of patents: Lessons from recent u.s. patent reform', *Journal Of Economic Perspectives* **16**(2), 131–154. available at http://ideas.repec.org/a/aea/jecper/v16y2002i2p131-154.html.

Gandal, N. (1994), 'Hedonic price indexes for spreadsheets and an empirical test of the network externalities hypothesis', *RAND Journal Of Economics* (1), 160–170.

Gehring, R. A. (2006), 'The institutionalization of open source', *Poiesis & Praxis: International Journal Of Technology Assessment And Ethics Of Science* **4**(1), 54–73.

Ghosh, R. A., Glott, R., Krieger, B. & Robles, G. (2002), Free/libre and open source software: Survey and study (floss) part 4: Survey of developers, Technical report, International Institute of Infonomics, University of Maastricht.
URL: *http://www.infonomics.nl/FLOSS/report/*

Graham, S. & Somaya, D. (2004), The use of patents, copyrights and trademarks in software: Evidence from litigation, *in* OECD, ed., 'Patents, Innovation And Economic Performance', OECD, Paris.

Gröhn, A. (1999), *Netzwerkeffekte und Wettbewerbspolitik. Eine Ökonomische Analyse Des Softwaremarktes*, Mohr Siebeck, Tübingen.

Hart, O. & Moore, J. (1990), 'Property rights and the nature of the firm', *Journal Of Political Economy* **98**(6), 1119–1158.

Hawkins, R. E. (2004), 'The economics of open source software for a competitive firm', *Netnomics* **6**(2), 103–117.

Heller, M. (1998), 'The tragedy of the anticommons: Property in the transition from marx to markets', *Harvard Law Review* (3), 621–688.

Henkel, J. (2006), 'Selective revealing in open innovation processes: The case of embedded linux', *Research Policy* **35**(7), 953–969.

Henkel, J. & Maurer, S. M. (2007), 'The economics of synthetic biology', *Molecular Systems Biology* **3**(Article number: 117).

Jensen, C. & Scacchi, W. (2007), Role migration and advancement processes in ossd projects, International Conference On Software Engineering, To Appear (29), Minneapolis, MN, USA.

Katz, M. L. & Shapiro, C. (1994), 'Systems competition and network effects. (symposia network externalities)', *Journal Of Economic Perspectives,* **8**(2), 93–115.

Kooths, S., Langenfurth, M. & Kalwey, N. (2003), *Open-Source Software: An Economic Assessment*, Vol. 4 of *MICE Economic Research Studies*, Muenster Institute For Computational Economics, Münster.

Langlois, R. N. (2002), 'Modularity in technology and organization', *Journal Of Economic Behavior & Organization* **49**(1), 19–37.

Lerner, J., Pathak, P. A. & Tirole, J. (2006), 'The dynamics of open-source contributors', *The American Economic Review* **96**(2), 114–118.

Lerner, J. & Tirole, J. (2002), 'Some simple economics on open source', *Journal Of Industrial Economics* **50**(2), 197–234.

Liebowitz, S. J. & Margolis, S. E. (1994), 'Network externality: An uncommon tragedy.', *Journal Of Economic Perspectives* **8**(2), 133–150.

Liebowitz, S. J. & Margolis, S. E. (2002), Network effects, *in* M. Cave, S. Majumdar & I. Vogelsang, eds, 'Handbook Of Telecommunications Economics', Vol. 1, pp. 76–94.

Maurer, S. M. (2008), 'Open source biology: Finding a niche (or maybe several)', *UMKC Law Review* **76**(2).

McGowan, D. (2001), 'The legal implications of open source software', *Illinois Law Review* (1), 241–304.

Nüttgens, M. & Tesei, E. (2000), Open Source: Marktmodelle und Netzwerke, Veröffentlichungen des Instituts für Wirtschaftsinformatik, Saarbrücken.

Osterloh, M., Rota, S. & von Wartburg, M. (2001), Open source - new rules in software development, Technical report.

Pasche, M. & von Engelhardt, S. (2004), Volkswirtschaftliche Aspekte der Open-Source-Softwareentwicklung, Jenaer schriften zur wirtschaftswissenschaft.

Picot, A., Dietl, H. & Franck, E. (2005), *Organisation: eine ökonomische Perspektive*, 4., überarb. und erw. aufl edn, Schäffer-Poeschel.

Quah, D. (2003), Digital goods and the new economy, CEP Discussion Papers 563, London School of Economics, London.

Romberg, T. (2003), Herstellerübergreifende Wiederverwendung von Komponenten, *in* 'Handbuch Zur Komponentenbasierten Softwareentwicklung', Frauenhofer-Institut Für Experimentelles Software Engineering / Forschungszentrum Informatik.

Rossi, C. & Bonaccorsi, A. (2006), Intrinsic motivations and profit-oriented firms in open source software: Do firms practise what they preach?, *in* J. B. Schröder & P. J. H., eds, 'The Economics Of Open Source Software Development', Elsevier, pp. 84–109.

Rossi, M. A. (2006), Decoding the free/open source software puzzle: A survey of theoretical and empirical contributions, *in* J. Bitzer & P. Schröder, eds, 'The Economics Of Open Source Software Development', Elsevier, pp. 15–55.

Shy, O. (2001), *The Economics Of Network Industries.*, Cambridge University Press, Cambridge.

von Engelhardt, S. (2006), Die ökonomischen Eigenschaften von Software, Jenaer Schriften zur Wirtschaftswissenschaft.

von Engelhardt, S. (2008), The economic properties of software, Jena Economic Research Papers 2008-045, Friedrich-Schiller-University Jena and Max-Planck-Institute of Economics.

von Hippel, E. (2005), Open source software projects as user innovation networks - no manufacturer required, *in* J. Feller, B. Fitzgerald, S. A. Hissam & K. R. Lakhani, eds, 'Perspectives On Free And Open Source Software', MIT Press, Cambridge, Mass. [u.a.].

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.

Weber, S. (2004), *The Success Of Open Source*, Harvard University Press.

Wendel de Joode, R. V., Bruijn, J. A. d. d. & Eeten, M. J. G. V. (2003), *Protecting The Virtual Commons – Self-Organizing Open Source And Free Software Communities And Innovative Intellectual Property Regimes*, T.M.C. Asser Press, The Hague.

Westarp, F. v. (2003), *Modeling Software Markets*, Physica-Verlag, Heidelberg [u.a.].

White, A. G., Abel, J. R., Berndt, E. R. & Monroe, C. W. (2004), Hedonic price indexes for personal computer operating systems and productivity suites, NBER Working Papers 10427, National Bureau of Economic Research, Inc.