

Open Source – Complexity, Cooperation and Credits*

Matthias Bärwolff

June 22, 2006

Abstract

Apart from the commonly cited reasons for the success of open source – namely the institutional framework comprising copyright based permissive licensing, the availability of source code, and some social norms that seem to guide much activity – there are two less often cited yet profound reasons for the viability of open source: first, the impossibility of drawing complete contracts over both open source development and licensing, and, second, the relative efficiency of a non-monetary accounting of credits in such circumstances.

1 Introduction

From what we have seen thus far in the series of lectures we know that good governance creates an important framework for the progress and viability of a society ([Lutterbeck 2006](#)). In fact good governance is a more important prerequisite for economic welfare than strictly market conforming institutions alone ([Wagener 2004](#)). This observation is all the more important as an unfettered belief in market forces and market forces alone shapes much of the discussion about the proper role of politics in our society.

While good governance is important on a macro level with regard to the society as a whole, it is no doubt important on a firm level, too. [Malone \(2004\)](#) notes that the Taylorian vision of firms as efficient command-and-control structures is no longer tenable:

In the old world of large-scale, mostly routine production, taking maximum advantage of everyone's intelligence and creativity wasn't critical, and the top-down, command-and-control management style was usually quite effective. But as organizations become more decentralized, as knowledge work comes to dominate

*Part of the lecture notes for the summer 2006 course "Information Rules 2", at department "Informatik und Gesellschaft", Technische Universität Berlin

the economy, and as innovation becomes increasingly important, taking advantage of people's true intelligence and creativity will become one of the most critical capabilities of successful businesses. (p. 153)

For the software industry the problem alluded to by Malone is particularly pronounced: software is a most complex good, there are severe limits to the rationality of the players involved, thus incomplete contracts are inevitable. And due to the public good nature of software (non-rival, and due to zero duplication costs effectively inexhaustible, and non-excludable) there is a structural conflict between the ends of private parties involved on the supply side and those on the demand side.

Open Source has emerged as a very promising approach to the problems inherent in software production and dissemination (Bessen 2001; Assay 2005; Gehring 2005) and may guide the way toward better software on a broad scale. This paper shall proceed as follows. First, we look briefly at the institutions that have come to govern much of the current activities concerned with open source. Second, we address the question of why private parties choose to commit substantial resources to open source development that inevitably benefits third parties without direct compensation.

2 Institutions of Open Source

The single most important institution, the rule that is the most binding and enforceable of all, is the chosen licence for a given open source software.¹ In order to qualify for an approval by the Open Source Initiative² a licence will have to meet a number of criteria:³

- There must be no restrictions on redistribution.
- There must be source code available with the software.
- Modifications and derived works must be allowed to be implemented and redistributed.

¹I do not intend to elaborate too extensively on the institutions of open source. Refer to Gehring (2005) for an introductory account of open source institutions. The notion of institutions is captured neatly by Ostrom (2005): "Institutional rules are often self-consciously crafted by individuals to change the structure of repetitive situations that they themselves face in an attempt to improve the outcomes that they achieve" (p.18).

²The Open Source Initiative (www.opensource.org) was founded by Eric S. Raymond who has been a central figure in the highly successful late 1990s efforts of gaining more recognition in the business community for the merits of a movement then called "free software".

³For the detailed list of criteria see: <http://opensource.org/docs/definition.php>.

- There must be no discrimination against persons, fields of endeavour, applications with other software or technology.

The legal validity of licences following such standards is widely accepted today. A number of court cases have enforced vital provisions of open source licences.⁴ Hence, licences such as the popular GNU General Public Licence (GPL)⁵ and the Berkeley Software Distribution Licence (BSD)⁶ are effective means of enforcing a basic set of rules for open source.

The licencing of open source software effects two institutions that coincide neatly with the inherent characteristics of digital information goods: availability of source code and unfettered freedom to copy, modify and redistribute. This in turn entails effectively zero procurement costs and thus a socially optimal distribution of software. Any positive utility on the demand side will be satisfied, and there is no welfare loss due to monopoly pricing beyond marginal cost (Gehring 2005). In sum, in todays era of broadband it is pretty much an inevitable rule that open source software is practically available free of any charge. Selling shiny boxes with open source software is not a viable business model anymore.

Beyond those rather obvious institutions are those "soft institutions" that require a more elaborate treatment and are prone to misjudgment and sustained debate. Sure, there may be said to be social norms of voluntary cooperation and according punishment of non-contributors that guide some of the activities (Osterloh et al. 2004) but such approaches fail to explain why predominantly extrinsically motivated parties such as modern corporations would want to join such efforts. To put it in economic terms: why not realise gains from trade where they occur? Any serious approach to describing the motivations behind open source must duly account for this question.

3 The Motivation Puzzle

There is a great wealth of inquiries into the motivations of open source contributors (Lerner and Tirole 2000; Ghosh et al. 2002; Osterloh et al. 2004; Hetmank 2005; Bärwolff 2006; Priddat 2006). Thus we know that most contributors do benefit from their efforts in one way or another. Of course, it is circular and thus very much unscientific to state that the individual utility makes the efforts worthwhile if we include all sorts of untestable utilities such as joy and fulfillment. However, Hars and Ou (2001) have noted that the rationality thesis holds even if we dispense with such evasive utilities: the tangible individual utility coincides strongly with the efforts employed.

⁴See e. g. Metzger (2004) and Ebinger (2005).

⁵<http://www.gnu.org/copyleft/gpl.html>.

⁶The BSD licence is the second most important licence behind the GPL. It does allow for modifications but does not require one to distribute the modified source code. It is thus more suitable for commercial uses of open source software.

Those developers whose contributions are most valuable either get payed outright by their employers or benefit indirectly from the sale of complementary assets such as services and customisation. They also create a strong signal for prospective employers that may be translated directly into current benefits ([Lerner and Tirole 2000](#)).

The same holds for commercial entities who may reasonably be assumed to behave mostly rational. A number of those create substantial revenues from selling services and products related to their often heavy involvement in open source software projects ([Bärwolff 2006](#)).

However, while we see private parties generating sufficient benefit to motivate their contributions to open source we still do not know why they do not prefer to realise in money terms the gains from trade they give rise to. The following sections put forward a number of approaches to explain for this.

4 Complexity and Incomplete Contracts

[Bessen \(2005\)](#) makes an important observation about the edge that open source has over proprietary software: it provides a means of coping with the difficulties of contracting over the complexity of software. Others have pointed out that there may be benefits from adopting an open source software development model due to the benefits from openness and redundancy ([Benkler 2002](#); [Hippel 2005](#)). However, such approaches do not conclusively answer the question why extrinsically motivated agents choose to “let go of their assets”. In this regard [Bessen](#) gives us a firmer clue:

Contracting over software generally has this difficulty: the software code itself is the complete description of how the software will function in every circumstance and, consequently, writing a contract that covers every contingency costs roughly the same as writing the software itself. Practical contracts for software will thus not completely specify all details, interactions and contingencies. This difficulty arises from the complexity of software. Typical software products have large numbers of features and the number of possible interactions between these features can be astronomical. ([2005](#), p.2)

Bessen posits that there are basically three solutions to reducing the significant transaction costs involved with complexity and incomplete contracts:

- Providing a large set of features in one software product, thus covering as wide a range of needs as possible to allow for economies of scale.
- Providing an application programming interface (API) and possibly toolkits in order to avoid complexity and allow for enhancements and customisations by or for the individual customer with his specific needs.

- Going open source, thus creating a commons that grows feature by feature whenever someone has sufficient incentive to provide a specific feature he is missing. Open source thus works much like the API solution with the key difference that the code is available to anyone without discrimination.

The three approaches, all of which we find ample examples for in the market, have their respective advantages and problems. Thus they are compliments rather than alternatives.⁷ Generally, in the model of [Bessen](#) open source will not per se be the superior solution to solving the problem of socially efficient software production and distribution. Open source makes most sense where there are highly specialised needs:

FOSS [open source software] does not displace a pre-packaged software monopolist, although the monopolist will charge lower prices and will lose API sales. Instead, pre-packaged software is sold for simpler applications used by large numbers of customers. Firms with specialized needs and more complex applications use FOSS. ([Bessen 2005](#), p. 3)

It is worth noting that while the three approaches are no direct substitutes for one another, they are not mutually exclusive and may well be leveraged at the same time, thus further reducing the costs from complexity. Arguably, the most successful software projects employ all of the three approaches – open source, API, and pre-packaging – alongside one another. For example, Red Hat, the world’s most successful commercial Linux distributor, co-develops all of the software it sells to its customers as open source in their respective communities under the GPL licence. Thus it provides not only a software that may be modified but a Linux API that is accessible

⁷As for the third solution – open source–, one problem is particularly severe in that it limits the scope of its application: the effort required to modify the software in order to fit ones needs may exceed the value of the resulting change. Just being open source does not necessarily entail easy modifiability. The code may be monolithic, technically flawed, and undocumented, as well as an API of a proprietary software may be undocumented, inconsistent, and flawed. Add to this the cost of making a reasonably informed judgement of costs to be due, the costs due to risk aversion, and resulting transaction costs, e. g. from outsourcing the development work, and the open source solution loses a lot of its appeal compared to the other two solutions [Bessen](#) identifies.

[Baldwin and Clark \(2006\)](#) remark that it is primarily two factors that make for the viability of open source: modularity and option value. They put it:

In the context of a voluntary collective process, modular architectures with high option value will elicit higher levels of effort and participation in equilibrium than monolithic, low-option-value architectures. (p. 28)

See also [Gutsche \(2005\)](#) who analyses the conditions under which open source software attracts sufficiently large communities, and arrives at a similar conclusion. However, in both accounts it remains largely unclear how best to foster such conditions.

without any discrimination whatsoever. On the other hand, customers may buy from Red Hat a fully featured and stable system with extensive support and financial backing, something that is not very much different from buying a pre-packaged system from a proprietary vendor ([Knoblich 2006](#)).⁸

The benefits on the customer side are rather obvious: potentially more control, more flexibility, better software, lower prices. On the production side, too, there are sizable benefits with respect to taming the complexity of software and the contingencies that may occur when applied in the field: not only the development efforts, but the whole (and usually very expensive) beta phase of a software is practically being played out in the commons, mostly absent any legally binding contractual relationships. Due to negligible communication and transaction costs the development efforts and the test cycles are being spread over all parties with sufficient incentives to join. It is very likely that those efforts, in particular those related to testing, are thus being allocated more efficiently than in the case of proprietary software development where the organisation of the beta phase is typically a very cumbersome process precisely because of the need for contractual agreements with third parties. After completion of beta testing in an open source manner stable software may be sold to customers along with support and training. In fact, such transactions then very much resemble the provision of a proprietary software.⁹

The case of Red Hat referred to above shows us the favourable interdependencies in the coupling of all three approaches suggested by [Bessen \(2005\)](#). It is possible that the Red Hat business model is not only a very profitable one – effectively making use of resources beyond the firm and their payroll ([Grand et al. 2004](#)) – but one that solves some of the problems of open source alluded to above.¹⁰ The most efficient means of solving the complexity problem of software may not be *one* of the three solutions open source, API, or pre-packaging, but the simultaneous use of *all* three. That way we may arrive at software that satisfies the needs of customers in the market and

⁸A consumer who needs to buy such a package will be indifferent about the prices of the individual components as long as the package price is the same ([Shapiro and Varian 1999](#); [Farrell and Shapiro 2004](#)).

⁹Contracts over such packages may be designed such that the risks from contractual contingencies are being reduced for the vendor as is typically the case in proprietary end user licencing agreements (EULA) and their "no warranty" provisions. The vendor may, however, assume further risks such as certifications (promising the fitness of a particular software to meet certain specifications) and guarantees for which he then may charge accordingly.

In any case, no matter who ultimately assumes the risk in subsequent contracts over complementary goods to open source software it is likely that the aggregate risk from the complexity of software is being reduced by an open source approach to developing software. Arguably, both risks and costs are being spread to where they are most efficiently minimised.

¹⁰See above footnote 7 and [Baldwin and Clark \(2006\)](#).

at the same time provides an efficient common ground for modifications and enhancements by all interested parties along the lines suggested by [Baldwin and Clark \(2006\)](#) and [Gutsche \(2005\)](#).

5 The Superiority of Tacit Credit Accounting

The open source way of software development may be more efficient than the proprietary one, yet one might still ask, and rightly so, why developers or firms involved not capitalise directly on the gains of trade that they create. The answer has to do partly with what has been said above – most do ultimately benefit –, and partly with what might appropriately be called the superiority of tacit accounting of credits.

It is rather obvious that whenever there are gains from trade that can feasibly be arrived at there is a possible compensation that would make the transaction Pareto efficient. If the costs from bringing such compensation about exceed the gains from trade at stake we have two basic market oriented possibilities.¹¹ We may dispense with the transaction all together, or, alternatively, find another way of accounting for the credits and debts that accrue in the process of open source development.

Such accounting actually takes place in open source development, and it is well understood that its existence is important for the success of open source. Tacit accounting may, in fact, be understood as a central institution in open source projects. Without such an institution no one could ever claim any credits in open source development, something that, in fact, frequently happens as we have seen above. It may at first glance seem a little far-fetched to liken credits in open source projects – entries in the credits-file, on the project website, in mailing lists, forums, etc. – to credits in a capitalist economic system. Yet, this is one vital function of credits in open source projects: to account for achievements in those projects. [Innes \(1914\)](#) has noted a long time ago that the essence of commerce is not money, as is sometimes assumed naively, but credit and credit alone. Such credits in open source projects may be redeemed or extinguished just like the more conventional sorts of credit such as banknotes or bills of exchange. Note that money, too, is nothing but a credit to the society as a whole with no intrinsic value whatsoever:

[M]oney is only a claim upon society. Money appears, so to speak, as a bill of exchange from which the drawee is lacking. [...] The liquidation of every private obligation by money means that the community now assumes this obligation to the creditor. ([Simmel](#)

¹¹There is, of course, a third possibility: command-and-control. However, such approach suffers from well-known deficiencies since the knowledge of individual utilities evades the party that has to make the resource allocations ([Hayek 1945](#)).

1978, pp. 177–178)

While no doubt money as the most easily transferable form of credit plays an important role in the history of capitalism (Ingham 2004) it is by no means said that it must be the most efficient form of credit in all circumstances. In particular, the quantification of credits in but one universal abstract measure, be it Euros, Dollars, or any sufficiently stable currency for that matter, poses non-trivial valuation problems. At times it is more appropriate to pose a credit in non-monetary terms.¹² While such an approach might be mimicked by drafting an appropriate contract doing so would entail considerable transaction costs, thus rendering the transaction futile. In an economic system, however, where trust and reciprocity are deeply entrenched (Gouldner 1960; Bolton and Ockenfels 2000; Falk and Fischbacher 2000; Diekmann 2004) a system of tacit accounting economises on the transaction costs inevitably associated with the maintenance of a monetary system. Of course, owing to their tacit nature some credits may never be redeemed in effect making them worthless. But such risk is not limited to credits in open source projects and may be discounted appropriately. Thus we will have to add such discount and a risk premium to the costs of operating a tacit credit accounting system. We are left to guess how large this cost is, but still, it appears that in the real-world instances of Linux and other highly distributed development efforts those costs are substantially lower than that of running a money based system of credit accounting.

6 Conclusion

The tacit credit accounting approach provides us with a unified means of understanding and modelling a wide array of open source development efforts and the incentives behind those efforts. Being so general it runs the risk of becoming vague and tautological, but such criticism may be applied to other models such as transaction costs (Coase 1937), too. At the least, a credit approach to the development, the incentives behind, and the actual structure of open source projects gives us a lens through which to look upon seemingly unconnected strands of open source activities, and formulate further relevant research questions. Further valuable insights and policy implications could be yielded by future research along the following lines:

- Comparative empirical analysis of transaction cost structures in software development
- Nature of credits in open source projects, redemption rates, discounts and costs due to risk aversion

¹²Think of things that are not even buyable such as status, reputation, a public office, etc.

It remains to be seen whether the credit approach allows for further useful and original insights into the ever more important reality of open source.

A second conclusion that we might express in a preliminary manner is that dedicated efforts by private and extrinsically motivated parties may help to build bridges between dispersed open source development communities and the needs of end users in the market place. Such efforts will further both private and public ends in that we create software that is usable for less technically versed customers, easily modifiable and enhanceable on the basis of technically excellent APIs and code architecture, and capable of efficiently pooling distributed development efforts of a larger community.

References

- Assay, N. M. (2005). Open source and the commodity urge: Disruptive models for a disruptive development process. In C. DiBona, D. Cooper, and M. Stone (Eds.), *Open Sources 2.0: The Continuing Evolution*. London: O'Reilly.
- Baldwin, C. and K. Clark (2006). The architecture of cooperation: Does code architecture mitigate free riding in the open source development model? *Management Science*. Forthcoming, special issue on open source software. Available at <http://www.people.hbs.edu/cbaldwin/DR2/BaldwinArchPartAll.pdf>.
- Benkler, Y. (2002). Coase's penguin, or, Linux and The nature of the firm. *Yale Law Journal* 112(3), 369–446.
- Bessen, J. E. (2001). Open source software: Free provision of complex public goods. *Research on Innovation* 7(2).
- Bessen, J. E. (2005). Open source software: Free provision of complex public goods. This paper is a substantially revised version of the 2001 paper with the same title. Available at SSRN: <http://ssrn.com/abstract=588763>.
- Bolton, G. and A. Ockenfels (2000). A theory of equity, reciprocity and competition. *American Economic Review* 100(1), 166–93.
- Bärwolff, M. (2006). Tight prior open source equilibrium. *First-Monday* 11(1). http://www.firstmonday.org/issues/issue11_1/barwolff/index.html.
- Coase, R. H. (1937). The nature of the firm. *Economica* 4, 386–405.
- Diekmann, A. (2004). The power of reciprocity. *Journal of Conflict Resolution* 48(4), 487–505.
- Ebinger, T. (2005). Tragen die Juristen Open-Source-Software zu Grabe? – Die GNU GPL vor Gericht. In B. Lutterbeck, M. Bärwolff, and

- R. A. Gehring (Eds.), *Open Source Jahrbuch 2005. Zwischen Softwareentwicklung und Gesellschaftsmodell*, pp. 249–269. Berlin: Lehmanns Media. <http://www.opensourcejahrbuch.de/2005>.
- Falk, A. and U. Fischbacher (2000). A theory of reciprocity. Working Paper 6, Institute for Empirical Research in Economics, University of Zurich.
- Farrell, J. and C. Shapiro (2004). Intellectual property, competition, and information technology. In H. R. Varian, J. Farrell, and C. Shapiro (Eds.), *The Economics of Information Technology - An Introduction*, pp. 49–86. Cambridge: Cambridge University Press.
- Gehring, R. (2005). The Institutionalization of Open Source. *Poiesis und Praxis* 4(1), 54–73.
- Ghosh, R. A., R. Glott, B. Krieger, and G. Robles (2002). Floss final report - part 4: Survey of developers. In *Free/Libre and Open Source Software: Survey and Study*. International Institute of Infonomics, University of Maastricht and Berlecon Research GmbH. http://www.infonomics.nl/FLOSS/report/FLOSS_Final4.pdf.
- Gouldner, A. W. (1960). The norm of reciprocity: A preliminary statement. *American Sociological Review* 25(2), 161–78.
- Grand, S., G. von Krogh, D. Leonard, and W. Swap (2004). Resource allocation beyond firm boundaries: A multi-level model for open source innovation. *Long Range Planning* 37, 591–610.
- Gutsche, J. (2005). The evolution of open source communities. *Topics in Economic Analysis & Policy* 5(1). <http://www.bepress.com/bejeap/topics/vol5/iss1/art2/>.
- Hars, A. and S. Ou (2001). Working for free? motivations of participating in open source projects. Proceedings of the 34th Hawaii International Conference on System Sciences, <http://csdl.computer.org/comp/proceedings/hicss/2001/0981/07/09817014.pdf>.
- Hayek, F. A. (1945). The use of knowledge in society. *American Economic Review* 35(4), 519–530.
- Hetmank, M. (2005). Open-Source-Software als Signal. In B. Lutterbeck, R. A. Gehring, and M. Bärwolff (Eds.), *Open Source Jahrbuch 2005. Zwischen Softwareentwicklung und Gesellschaftsmodell*, pp. 177–184. Berlin: Lehmanns Media. <http://www.opensourcejahrbuch.de/2005/>.
- Hippel, E. (2005). Democratizing Innovation. Available at SSRN: <http://ssrn.com/abstract=712763>.
- Ingham, G. (2004). The emergence of capitalist credit money. In L. R. Wray (Ed.), *Credit and State Theories of Money: The Contributions*

- of *A. Mitchell Innes*, pp. 173–222. Cheltenham: Edward Elgar.
- Innes, A. (1914). The credit theory of money. *Banking Law Journal* 31, 151–168.
- Knoblich, W. (2006). Erfolgreich mit Open Source – Das Red-Hat-Open-Source-Geschäftsmodell. In B. Lutterbeck, M. Bärwolff, and R. A. Gehring (Eds.), *Open Source Jahrbuch 2006. Zwischen Softwareentwicklung und Gesellschaftsmodell*, pp. 155–164. Berlin: Lehmanns Media. <http://www.opensourcejahrbuch.de/2006>.
- Lerner, J. and J. Tirole (2000). The simple economics of open source. *Journal of Industrial Economics* 52, 197–234. NBER Working Paper, Nr. 7600.
- Lutterbeck, B. (2006). Good and bad governance: Strukturen, Koordination, Kooperation und Evolution. Lecture notes for a lecture held at Technische Universität Berlin, summer 2006. http://ig.cs.tu-berlin.de/lehre/s2006/ir2/v1_etc/date-1/IR2_Hinweise-Literatur_Vorlesungen1-3.pdf.
- Malone, T. (2004). *The Future of Work: How the New Order of Business Will Shape Your Organization, Your Management Style, and Your Life*. Harvard Business School Press.
- Metzger, A. (2004). Wirksamkeit einer GPL-Lizenz – LG München I, Urteil vom 19.5.2004, mit Anmerkungen von Hoeren und Metzger. *Computer und Recht* (10), 774–779.
- Osterloh, M., S. Rota, and B. Kuster (2004). Open-Source-Softwareproduktion: Ein neues Innovationsmodell? In B. Lutterbeck and R. A. Gehring (Eds.), *Open Source Jahrbuch 2004. Zwischen Softwareentwicklung und Gesellschaftsmodell*, pp. 121–138. Berlin: Lehmanns Media. <http://www.opensourcejahrbuch.de/2004>.
- Ostrom, E. (2005). *Understanding Institutional Diversity*. Princeton University Press. Sample chapter available at <http://www.pupress.princeton.edu/chapters/s8085.pdf>.
- Priddat, B. P. (2006). Open Source als Produktion von Transformationsgütern. In B. Lutterbeck, M. Bärwolff, and R. A. Gehring (Eds.), *Open Source Jahrbuch 2006. Zwischen Softwareentwicklung und Gesellschaftsmodell*, pp. 109–121. Berlin: Lehmanns Media. <http://www.opensourcejahrbuch.de/2006>.
- Shapiro, C. and H. R. Varian (1999). *Information Rules: A Strategic Guide to the Network Economy*. Harvard: Harvard Business School Press.
- Simmel, G. (1978). *The Philosophy of Money*. London: Routledge. First published 1907.

Wagener, H.-J. (2004). Good governance, welfare, and transformation. *The European Journal of Comparative Economics* 1(1), 127–143. <http://eaces.liuc.it>.