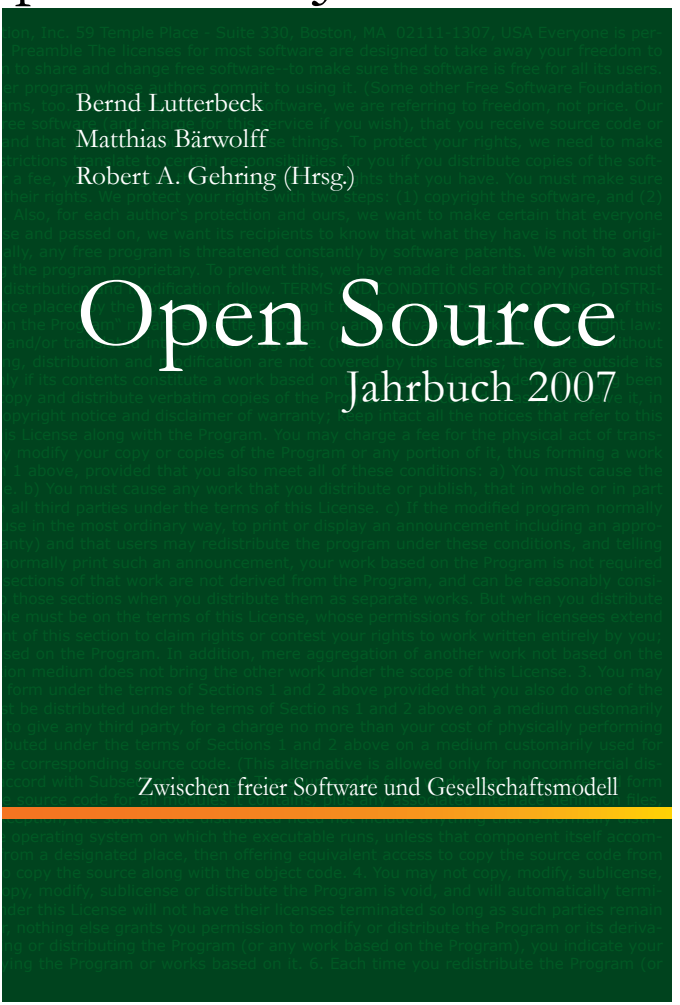
Dieser Artikel ist Teil des

Open Source Jahrbuchs 2007



erhältlich unter www.opensourcejahrbuch.de.

Die komplette Ausgabe enthält viele weitere interessante Artikel. Sie können diesen und andere Artikel im Open-Source-Jahrbuch-Portal kommentieren oder bewerten: www.opensourcejahrbuch.de/portal/. Lob und Kritik sowie weitere Anregungen können Sie uns auch per E-Mail mitteilen.

Entwicklung von Open-Source-Software: Kostenrelevante Eigenschaften einer ungewöhnlichen Organisationsform

JÜRGEN BITZER UND PHILIPP J. H. SCHRÖDER

In der Auseinandersetzung zwischen Vertretern der kommerziellen Softwareindustrie und der Open-Source-Software-Gemeinde werden häufig Argumente zur Effizienz der unterschiedlichen Entwicklungsmodelle ins Feld geführt. Insbesondere Erstere nehmen für sich in Anspruch, das effizientere Entwicklungsmodell zu vertreten und sehen daher in der Konkurrenz durch Open-Source-Software (OSS) eine Gefahr für die Innovationsfähigkeit der Softwarebranche. Eine Analyse kostensparender und -verursachender Eigenschaften des OSS-Entwicklungsmodells zeigt jedoch, dass an der Auffassung, dass das kommerzielle Entwicklungsmodell effizienter sei, erhebliche Zweifel angebracht sind.

Schlüsselwörter: Open-Source-Software · Forking-Thread-Effekt · Effizienz · Software-Entwicklungsmodell · Kostenstruktur

1 Einleitung

Das Auftauchen von marktfähiger Open-Source-Software (OSS) hat die Softwarebranche grundlegend verändert. Nicht nur, dass in vormals hoch konzentrierten Marktsegmenten durch OSS-Programme Wettbewerb entstanden ist (z. B. Linux bei Server-Betriebssystemen), vielmehr stellt der sich selbst organisierende Entwicklungsprozess von OSS traditionelle Entwicklungsmethoden in Frage.

Von Vertretern der Softwareindustrie wurde wiederholt die Frage aufgeworfen, ob der zunehmende Wettbewerb und der unkonventionelle Entwicklungsstil von OSS tatsächlich zu einer Verbesserung der Innovationsaktivität in der Softwarebranche führen oder im Gegenteil diesen vielleicht sogar zum Erliegen bringen könnte (Bitzer und Schröder 2006).

Während beim Wettbewerbsargument darauf hingewiesen wird, dass die Unternehmen Monopolrenten benötigten, um ihre Entwicklungsausgaben zu amortisieren – ein Argument, welches häufig auch im Zusammenhang mit der Pharmaindustrie angeführt wird –, zielt das zweite Argument darauf ab, dass der straff organisierte kommerzielle Entwicklungsstil dem von OSS überlegen sei.

Während das Wettbewerbsargument bereits ausführlich in der Literatur diskutiert wurde (z. B. Bitzer 2004; Bitzer und Schröder 2003; Economides und Katsamakas 2006; Casadesus-Masanell und Ghemawat 2006), fand das zweite Argument bisher nur begrenzt Niederschlag in der wissenschaftlichen Diskussion. Es impliziert, dass die kommerzielle Entwicklung von Software effizienter ist als die im Rahmen von OSS. Exemplarisch sei hier die von *Microsoft* finanzierte Studie des *Muenster Institute for Computational Economics (MICE)* an der Universität Münster zitiert:

> „Die Open-Source-Entwicklungsmethode muss [. . .] unter volkswirtschaftlichen Gesichtspunkten als weder effizient noch effektiv eingeordnet werden. Sie ist weniger effektiv als kommerzielle Modell, weil es keinen Mechanismus gibt der überprüft, ob die eingesetzte Entwicklerkapazität tatsächlich in eine sinnvolle Verwendung fließt. Sie ist nicht effizient, weil der Einsatz im Verhältnis zum Ergebnis unangemessen hoch sein kann." (Kooths et al. 2003, S. 71 f.)

Eine nähere Betrachtung zeigt, dass eine Effizienzanalyse im herkömmlichen Sinne – also das Verhältnis von Input zu Output – beim Vergleich der zwei sich gegenüberstehenden Entwicklungsmodelle nicht möglich ist. Unser Papier wählt daher eine andere Strategie auf der Suche nach der Antwort, ob eines der beiden Entwicklungsmodelle effizienter ist. Ziel unserer Analyse ist es, die kostenrelevanten Unterschiede in der Organisation des Entwicklungsprozesses von kommerzieller Software und OSS herauszuarbeiten.

Es soll bereits an dieser Stelle darauf hingewiesen werden, dass unser Papier keine endgültige Antwort auf die aufgeworfene Frage geben kann. Vielmehr besitzt unsere Analyse Erkundungscharakter, wobei die Frage nach der Effizienz der beiden Entwicklungsmodelle anhand kostenrelevanter Charakteristika der Organisationsform der Softwareentwicklung eruiert wird. Hierbei zeigt sich, dass wichtige Unterschiede zwischen dem kommerziellen und dem OSS-Entwicklungsmodell bisher noch nicht untersucht worden sind.

2 OSS-Entwicklung: eine ökonomische Perspektive

In der Auseinandersetzung zwischen Vertretern der kommerziellen Softwareindustrie und der Open-Source-Software-Gemeinde werden häufig Argumente zur Effizienz der unterschiedlichen Entwicklungsmodelle ins Feld geführt. Insbesondere Erstere nehmen für sich in Anspruch, das effizientere Entwicklungsmodell zu vertreten und

sehen daher in der Konkurrenz durch Open-Source-Software (OSS) eine Gefahr für die Innovationsfähigkeit der Softwarebranche.

Bei näherer Betrachtung zeigt sich allerdings, dass ein Effizienzvergleich schlichtweg nicht möglich ist, da für OSS keine Preise existieren. Dies bedeutet aber nicht, dass OSS keinen Wert besitzt oder keinen Nutzen stiftet. Dies ist allein schon daran zu erkennen, dass OSS genutzt wird. Weiterhin lässt sich theoretisch auch ein Wert für OSS bestimmen, indem die Zahlungsbereitschaft von OSS-Nutzern abgefragt wird. Dies würde die Ableitung einer Nachfragekurve und damit die Ermittlung der Konsumentenrente ermöglichen. Wie allerdings aus der Literatur zu öffentlichen Gütern hinlänglich bekannt ist, ist die Ermittlung der Zahlungsbereitschaft von Konsumenten öffentlicher Güter in der Praxis mit erheblichen Schwierigkeiten verbunden (z. B. Brookshire et al. 1982; Kahneman und Ritov 1994).[1] Es sei an dieser Stelle darauf hingewiesen, dass auch für kommerzielle Software der Gesamtnutzen nicht ermittelt werden kann, da die Nachfragekurve i. d. R. nur für einen bestimmten Preisbereich bekannt ist. So ist beispielsweise für fast alle Produkte weder der Prohibitivpreis, also der Preis bei dem die Nachfrage auf Null sinkt, noch die Nachfrage bei einem Preis von Null bekannt.

Es liegt somit auf der Hand, dass ein Effizienzvergleich im Sinne eines Produktivitätsvergleichs nicht möglich ist, da zur Ermittlung der Effizienz sowohl Output als auch Input benötigt werden. Somit bleibt als einziger Weg der Vergleich der Organisation des Entwicklungsprozesses in Bezug auf kostensparende und kostenerhöhende Eigenschaften. Anders ausgedrückt, unter der Annahme, dass ein identisches Stück Software entwickelt werden soll, wird untersucht, mit welcher Organisationsform der Softwareentwicklung diese Aufgabe kostengünstiger erledigt werden kann.

Eine Durchsicht der Literatur zeigt, dass die Unterschiede zwischen dem Entwicklungsmodell von kommerzieller Software und OSS häufig auf offensichtliche Kostenunterschiede reduziert werden. Das heißt, auf der einen Seite steht die kostenverursachende kommerzielle Entwicklung und auf der anderen Seite die „kostenlose" Entwicklung von OSS. Diese vereinfachte Unterscheidung greift natürlich zu kurz, da auch bei der Entwicklung von OSS (private) Kosten anfallen, nur dass diese nicht an den Nutzer weitergegeben werden. Berücksichtigt man ferner, dass zu den tatsächlich anfallenden privaten Kosten eines OSS-Entwicklers – wie z. B. Kosten für Cola, Pizza und Zigaretten – auch noch Opportunitätskosten entstehen, da er in der gleichen Zeit auch einen bezahlten Job übernehmen könnte, kann vereinfachend unterstellt werden, dass die Kosten pro Programmierstunde ähnlich sind.

Somit bleibt die Frage, ob die bei OSS-Projekten eingesetzte Organisationsform des Entwicklungsprozesses möglicherweise zu Effizienzgewinnen führt. Wie die fol-

1 Abgesehen von der fundamentalen Trittbrettfahrerproblematik zeigen experimentelle Resultate, dass die Präferenzstruktur, die durch die Bereitschaft freiwillige Beiträge zu zahlen abgebildet wird, von der durch eigentliche Kaufentscheidungen abgebildeten Präferenzstruktur abweicht (Kahneman und Ritov 1994).

genden Ausführungen zeigen werden, lassen sich sowohl kostensparende als auch kostenerhöhende Eigenschaften des Entwicklungsprozesses von OSS identifizieren.

Obwohl jedes OSS-Projekt seine Eigenarten in der Organisationsform aufweist, können doch stilisierte Eigenschaften identifiziert werden, die für die überwiegende Mehrheit der OSS-Projekte Gültigkeit besitzen. Weiterhin sollen im Folgenden nur die kostenrelevanten Besonderheiten skizziert werden, in welchen sich das OSS–Entwicklungsmodell grundlegend vom kommerziellen Software-Entwicklungsmodell unterscheidet.

3 Kostensparende Eigenschaften des OSS-Entwicklungsmodells

Der frei zugängliche Quellcode von OSS ermöglicht Interessierten, den Quellcode zu lesen, nachzuvollziehen und daraus zu lernen. Wissen kann somit ungehindert von einem Entwickler zum anderen diffundieren. Dementsprechend entstehen in der Entwicklergemeinde beträchtliche *Wissensspillovereffekte*. Eine ungehinderte Wissensdiffusion in und zwischen OSS-Projekten spart Entwicklungskosten, da auf bereits vorhandenes Wissen (z. B. Lösungsansätze und Erfahrungswissen) zurückgegriffen werden kann.[2] Im kommerziellen Entwicklungsmodell hingegen ist der Zugang zu den Entwicklungen anderer Teams im Unternehmen häufig durch Hierarchien und Organisationsstrukturen limitiert.

Weiterhin erhöht der frei zugängliche Quellcode von OSS die Motivation der Programmierer, denn die Nachvollziehbarkeit der Programmierschritte hat einen disziplinierenden Effekt: Das zusehende „Publikum" spornt den Programmierer an, seine Programme klar strukturiert und effizient zu schreiben. Ein besonders gelungenes Programm erhöht zum einen die Reputation innerhalb der Entwicklergemeinde (vgl. Bitzer et al. 2007; Raymond 1999*b*; Torvalds und Diamond 2001) und eröffnet ihm zum anderen die Möglichkeit, seine Programmierfähigkeiten potenziellen Arbeitgebern zu signalisieren (vgl. Leppämäki und Mustonen 2004*a*,*b*; Lerner und Tirole 2002; Raymond 1999*a*). Zusätzlich wird die Motivation der Programmierer durch die Freiwilligkeit der Beteiligung gesteigert. Bei ehrenamtlichen OSS-Entwicklern liegt es in der Natur der Sache, dass sie i. d. R. an Projekten arbeiten, an denen sie auch arbeiten wollen und deshalb ihre Arbeit „genießen" (vgl. Bitzer et al. 2007; Luthiger 2005). Diese aus den Eigenschaften der Software und der beteiligten Personen entstehenden, motivationserhöhenden Anreize müssen in einem Unternehmen entweder mittels monetärer Instrumente etabliert werden oder sind – wie beispielsweise im Falle der Motivation durch die „Freiwilligkeit" – gar nicht realisierbar.

Eine weitere kostensparende Besonderheit des OSS-Entwicklungsmodells ist die *ungehinderte Kooperation* zwischen den Programmierern. Da eine Vermarktung der Software nicht beabsichtigt ist, gibt es keine strukturellen Gründe, wie beispielsweise

2 Für eine theoretische Analyse vgl. Bitzer und Schröder 2005.

Geheimhaltung, die eine Kooperation innerhalb der Entwicklergemeinde behindern. Das heißt, sich gegenseitig zu helfen oder eine Arbeitsteilung zu vereinbaren wird nicht durch kommerzielle Interessen verhindert (vgl. Haefliger et al. 2005; Lakhani und von Hippel 2003). Komplementäre Fähigkeiten von Programmierern können somit besser genutzt werden und erhöhen daher die Effizienz bei der Entwicklung von OSS.

Auch der sogenannte Forking-Thread-Effekt (vgl. Bitzer und Schröder 2006) reduziert Kosten in der Entwicklung von OSS durch das Entstehen von demokratischen Entscheidungsstrukturen. Die freiwillige Teilnahme der Programmierer auf der einen Seite und die Drohung, das OSS-Projekt auf eigene Faust in eine andere Richtung weiterzuentwickeln, auf der anderen Seite, stellen sicher, dass die Entwicklergemeinde bei wichtigen Entscheidungen entsprechend demokratischen Prinzipien beteiligt wird. Hierdurch werden zwei Dinge gewährleistet: Erstens stehen technische Aspekte im Zentrum der Entscheidungen, in welche Richtung ein OSS-Projekt weiterentwickelt werden soll. Zweitens werden neue Softwareentwicklungen oder *bugfixes* umgehend implementiert. Der dadurch bestimmte Umgang mit vergangenen Entwicklungskosten ist radikal anders als der in kommerziellen Softwareunternehmen. Während diese konstant unter dem Druck stehen, bereits realisierte Investitionen in Softwareentwicklungen durch einen entsprechenden Markterfolg zu amortisieren, können in OSS-Projekten bereits geleistete (private) Entwicklungskosten ohne weiteres übergangen werden, sobald technisch bessere Alternativen vorliegen. Bei kommerziellen Softwareunternehmen hingegen müssen sich die Entwicklungskosten amortisieren, was dazu führen kann, dass in veraltete oder aussichtslose Technologien weitere Ressourcen investiert werden müssen, nur um nicht schon erbrachte Investitionen zu verlieren. Im Gegensatz dazu sind OSS-Projekte in der Lage, dem ökonomisch korrekten Prinzip von *bygones are bygones* zu folgen.[3] Es bleibt somit festzuhalten, dass der Forking-Thread-Effekt und die daraus entstehenden demokratischen Prinzipien Investitionen in technologische Fehlentwicklungen verhindern können.[4]

Eine weitere wichtige kostenreduzierende Eigenschaft des OSS-Entwicklungsprozesses ist die Wiederverwendung von Quellcode (vgl. Spinellis und Szyperski 2004; Haefliger et al. 2005). Ermöglicht durch die Lizenzbedingungen verschiedener OSS–Lizenzen können einzelne Zeilen oder sogar ganze Programmteile in anderen OSS–Projekten wiederverwendet werden. Diese Rückgriffmöglichkeit auf vorhandenen Quellcode senkt die Kosten der Entwicklung von OSS im Gegensatz zur Entwicklung von proprietärer Software. In letzterem Fall fällt die Nutzung von OSS-Quellcode aus,

3 In der politischen Ökonomie ist dieser Gegensatz unter dem Namen *Concord Fallacy* bekannt. Hier wurden jahrelang weitere Zuschüsse zur Entwicklung eines Überschallflugzeuges damit begründet, dass die bereits verbrauchten Ressourcen durch ein Einstellen des Projektes verloren gehen würden. In der Gesamtbetrachtung war das Projekt ein Verlust und das Prinzip von *bygones are bygones* hätte zu einer unmittelbaren Einstellung und somit der ökonomisch korrekten Verlustminimierung geführt.

4 Ein weiterer Kostenspareffekt entsteht auf der Seite der Nutzer, da diese schnell in den Genuss von neuen Entwicklungen und *bugfixes* kommen.

da ansonsten die gesamte Software als OSS zu veröffentlichen wäre.[5] Somit bleibt für Unternehmen höchstens die Wiederverwendung von Quellcode innerhalb des Unternehmens oder der Erwerb von Code von Dritten übrig.

Eine letzte kostensparende Eigenschaft des OSS-Entwicklungsprozesses ist der direkte Kontakt zwischen Entwickler und Nutzer. Oft sind Entwickler und Nutzer sogar ein und dieselbe Person, da ein zentrales Motiv für den Beginn von oder die Teilnahme an OSS-Projekten der Bedarf an einer speziellen Softwarelösung ist.[6] Doch selbst wenn Nutzer und Entwickler nicht ein und dieselbe Person sind, kann der Nutzer direkt Kontakt mit dem Entwickler aufnehmen, da dessen Name und E-Mail-Adresse öffentlich zugänglich sind. Die Kommunikation zwischen Entwickler und Nutzer wird also nicht durch Vertriebs- und Supportabteilungen oder Zwischenhändler beeinträchtigt. Die direkte Kommunikation ermöglicht eine schnelle Rückmeldung von Fehlern und Wünschen für neue Eigenschaften an den Entwickler. Zur Erhebung solcher Informationen müssen Unternehmen erhebliche Mittel investieren.

4 Kostenverursachende Eigenschaften des OSS-Entwicklungsprozesses

Natürlich können auch kostenverursachende Eigenschaften des OSS-Entwicklungsprozesses identifiziert werden. Beispielsweise besteht die Gefahr von *Doppelentwicklungen*. Hierbei geht es um das Problem, dass die Entwicklungsarbeit zwischen verschiedenen OSS-Projekten nicht koordiniert wird und somit Programme oder Programmteile unabhängig voneinander mehrfach entwickelt werden können. Weil das Programmieren einer Softwarelösung einen höheren Stellenwert in der Community hat als die Suche nach *prior art*, besteht weiterhin die Gefahr, dass Programmierer lieber ein neues Programm schreiben, anstatt ein schon vorhandenes zu verwenden oder zu modifizieren.

Weiterhin kann das Auseinanderbrechen eines OSS-Projektes in zwei Versionen, die in unterschiedliche Richtungen weiterentwickelt werden (*forking*), zu erheblichen Zusatzkosten im OSS-Entwicklungsprozess führen. So reduziert das Abspalten von Programmierern die Anzahl der Programmierer, die an den einzelnen Versionen arbeiten. Dies vermindert zum einen die Effekte der oben beschriebenen kostensparenden Eigenschaften (z. B. Diffusion von Wissen, ungehinderte Kooperation, Nutzung von komplementären Fähigkeiten) und zum anderen können Kosten für die Akquise von zusätzlichen Programmierern entstehen (Lattemann und Stiglitz 2006; Krishnamurthy und Tripathi 2006; Vujovic und Ulhøi 2006). In manchen Fällen müssen beispielsweise

5 Dies gilt zumindest für Programme, die unter der weit verbreiteten GPL-Lizenz stehen. Es gibt aber auch Open-Source-Lizenzen, wie etwa die *LGPL* oder die BSD-Lizenz, die eine Wiederverwendung des Codes in proprietärer Software erlauben. So basiert etwa der Kernel des Betriebssystems *Mac OS X* auf *NetBSD*, einem freien Betriebssystem, das unter der BSD-Lizenz steht.

6 Dieses Nutzer-Entwickler-Element wird als ein Hauptvorteil des OSS-Entwicklungsmodells angesehen (vgl. Kuan 2001; Franke und von Hippel 2003).

Bounties (Kopfgelder) ausgelobt werden, um drängende Probleme zu lösen (vgl. Krishnamurthy und Tripathi 2006). In anderen Fällen muss das Community-Management angepasst werden, damit neue Programmierer angelockt werden können (Lattemann und Stiglitz 2006; Vujovic und Ulhøi 2006). Im schlimmsten Fall jedoch entsteht eine Knappheit an Programmierern, die möglicherweise das ganze Projekt gefährdet. Weiterhin können inkompatible Standards entstehen, die eine Wiederverwendung von Quellcode verhindern.

5 Zusammenfassung und Ausblick

Die Ausführungen der vorangegangenen Abschnitte haben gezeigt, dass sich aus ökonomischer Sicht die Frage, ob das kommerzielle Software-Entwicklungsmodell oder das OSS-Entwicklungsmodell effizienter ist, nicht abschließend beantworten lässt. Auch wenn sich auf der Basis der vorangegangenen Diskussion keine allgemeingültige Aussage darüber treffen lässt, welches Entwicklungsmodell effizienter ist, können dennoch diverse Eigenschaften im OSS-Entwicklungsmodell identifiziert werden, die eindeutig zu Kostenersparnissen im Vergleich zum kommerziellen Software-Entwicklungsmodell führen. Besondere Bedeutung kommt hierbei den *Wissensspillovereffekten* innerhalb der OSS-Community, der Motivation der Programmierer, der ungehinderten Kooperation zwischen den Programmierern sowie der Wiederverwendung von Quellcode zu. Demgegenüber finden sich auch Eigenschaften, die Kosten verursachen, welche beim kommerziellen Entwicklungsmodell nicht entstehen. Beispiele hierfür sind redundante Entwicklungen oder *forking*.

Es liegt auf der Hand, dass nur eine Quantifizierung und Gegenüberstellung der kostensparenden und kostenerhöhenden Eigenschaften der zwei Entwicklungsmodelle Klarheit über Effizienzunterschiede bringen wird. Leider fehlen bislang entsprechende empirische Studien, die eine Quantifizierung der kostensparenden und kostenerhöhenden Effekte des OSS-Entwicklungsmodells vornehmen. Hieraus ergeben sich unmittelbar zukünftige Forschungsthemen. Beispielsweise ist bisher nicht empirisch untersucht worden, ob und in welchem Ausmaß Wissensdiffusion innerhalb einer OSS-Community stattfindet.[7]

Eine weitere Frage, die sich unmittelbar aus der Diskussion in unserem Papier ergibt, ist, ob sich das OSS-Entwicklungsmodell möglicherweise nur für bestimmte Softwarelösungen eignet. Sprich, gibt es Software, bei deren Entwicklung die kostensparenden Eigenschaften des OSS-Entwicklungsmodells nicht zum Tragen kommen?

Außerdem fokussierte sich unsere Diskussion auf die Entwicklungskosten der Herstellung von Software; ausgeblendet wurden hierbei Kosten, die auf der Seite der Nutzer entstehen. Um zu einer gesamtwirtschaftlichen Einschätzung zu gelangen, ist aber sicherlich weiterhin die *Total Cost of Ownership*, die neben den Anschaf-

7 Ein erster vielversprechender Forschungsansatz ist hier die Analyse sozialer Netzwerkstrukturen, siehe Xu et al. (2006).

fungskosten auch Folgekosten des Betriebs der Software umfasst, zu berücksichtigen. Auch in diesem Fall ist eine empirische Analyse dringend erforderlich. Eine gesamtwirtschaftliche Abschätzung der Auswirkungen von OSS auf die Innovationsaktivität bleibt somit zukünftigen empirischen Studien überlassen.

Literatur

Bitzer, J. (2004), 'Commercial versus Open Source Software: The Role of Product Heterogeneity in Competition', *Economic Systems* **28**(4), S. 369–381.

Bitzer, J., Schrettl, W. und Schröder, P. J. H. (2007), 'Intrinsic Motivation in Open Source Software Development', *Journal of Comparative Economics* . Zum Zeitpunkt des Drucks war dieser Artikel noch nicht veröffentlicht.

Bitzer, J. und Schröder, P. J. H. (2003), Competition and Innovation in a Technology Setting Software Duopoly, Discussion Paper 363, DIW Berlin. http://www.diw.de/deutsch/publikationen/diskussionspapiere/docs/papers/dp363.pdf [08. Dez 2006].

Bitzer, J. und Schröder, P. J. H. (2005), 'Bug-Fixing and Code-Writing: The Private Provision of Open Source Software', *Information Economics and Policy* **17**(3), S. 389–406.

Bitzer, J. und Schröder, P. J. H. (2006), The Impact of Entry and Competition by Open Source Software on Innovation Activity, *in* J. Bitzer und P. J. H. Schröder (Hrsg.), 'The Economics of Open Source Software Development', Elsevier Science, Amsterdam, S. 219–246.

Brookshire, D. S., Thayer, M. A., Schulze, W. D. und d'Arge, R. C. (1982), 'Valuing Public Goods: A Comparison of Survey and Hedonic Approaches', *American Economic Review* **72**(1), S. 165–177.

Casadesus-Masanell, R. und Ghemawat, P. (2006), 'Dynamic Mixed Duopoly: A Model Motivated by Linux vs. Windows', *Management Science* **52**(7), S. 1072–1084. http://opensource.mit.edu/papers/masanellghemawat.pdf [08. Dez 2006].

Economides, N. und Katsamakas, E. (2006), Linux vs. Windows: A Comparison of Application and Platform Innovation Incentives for Open Source and Proprietary Software Platforms, *in* J. Bitzer und P. J. H. Schröder (Hrsg.), 'The Economics of Open Source Software Development', Elsevier Science, Amsterdam, S. 207–218. http://www.stern.nyu.edu/networks/Economides_Katsamakas_Linux_vs._Windows.pdf [08. Dez 2006].

Franke, N. und von Hippel, E. (2003), 'Satisfying Heterogeneous User Needs via Innovation Toolkits: The Case of Apache Security Software', *Research Policy* **32**(7), S. 1199–1215. http://opensource.mit.edu/papers/rp-vonhippelfranke.pdf [08. Dez 2006].

Haefliger, S., von Krogh, G. und Spaeth, S. (2005), Knowledge Reuse in Open Source Software, *in* 'Proceedings of 38th Annual Hawaii International Conference on System Sciences', IEEE. http://csdl.computer.org/comp/proceedings/hicss/2005/2268/07/22680198b.pdf [08. Dez 2006].

Kahneman, D. und Ritov, I. (1994), 'Determinants of stated Willingness to pay for Public Goods: A Study in the Headline Method', *Journal of Risk and Uncertainty* **9**(1), S. 5–37.

Kooths, S., Langenfurth, M. und Kalwey, N. (2003), 'Open-Source-Software: Eine volkswirtschaftliche Bewertung', *MICE Economic Research Studies* **4**. http://mice.uni-muenster.de/mers/ [08. Dez 2006].

Krishnamurthy, S. und Tripathi, A. K. (2006), Bounty Programs in Free/Libre/Open Source Software (FLOSS), *in* J. Bitzer und P. J. H. Schröder (Hrsg.), 'The Economics of Open Source Software Development', Elsevier Science, Amsterdam, S. 165–184.

Kuan, J. W. (2001), Open Source Software as Consumer Integration into Production, Working Paper, Stanford Institute for Economic Policy Research. http://ssrn.com/abstract=259648 [05. Dez 2006].

Lakhani, K. R. und von Hippel, E. (2003), 'How Free and Open Source Software Works: Free User-to-User Assistance', *Research Policy* **32**, S. 923–943. http://www.cs.princeton.edu/courses/archive/fall02/frs129/readings/open source 2/lakhani and vonhippel on what keeps apache ticking.pdf [08. Dez 2006].

Lattemann, C. und Stiglitz, S. (2006), Coworker Governance in Open-Source Projects, *in* J. Bitzer und P. J. H. Schröder (Hrsg.), 'The Economics of Open Source Software Development', Elsevier Science, Amsterdam, S. 149–164.

Leppämäki, M. und Mustonen, M. (2004*a*), Signalling and Screening with Open Source Programming, Working Paper W-377, Helsinki School of Economics. http://helecon3.hkkk.fi/pdf/wp/w377.pdf [08. Dez 2006].

Leppämäki, M. und Mustonen, M. (2004*b*), Signalling with Externality, Working Paper W-376, Helsinki Center of Economic Research. http://helecon3.hkkk.fi/pdf/wp/w376.pdf [08. Dez 2006].

Lerner, J. und Tirole, J. (2002), 'Some Simple Economics of Open Source', *Journal of Industrial Economics* **50**(2), S. 197–234. http://uringmachine.org/opensource/papers/lerner2002.pdf [08. Dez 2006].

Luthiger, B. (2005), Fun and Software Development, *in* M. Scotto und G. Succi (Hrsg.), 'Proceedings of the First International Conference on Open Source Systems (11.-15. Juli 2005)', Genova, S. 273–278. https://www.foss.ethz.ch/people/lbenno/BLuthiger_Fun_SoftwareDevel_OSS2005.pdf [08. Dez 2006].

Raymond, E. S. (1999*a*), The Cathedral and the Bazaar, *in* 'The Cathedral and the Bazaar: Musings on Linux and Open Source from an Accidental Revolutionary', O'Reilly & Associates, Cambridge. http://catb.org/esr/writings/cathedral-bazaar/cathedral-bazaar/cathedral-bazaar.ps [05. Dez 2006].

Raymond, E. S. (1999*b*), Homesteading the Noosphere, *in* 'The Cathedral and the Bazaar: Musings on Linux and Open Source from an Accidental Revolutionary', O'Reilly & Associates, Cambridge. http://catb.org/esr/writings/cathedral-bazaar/homesteading/homesteading.ps [05. Dez 2006].

Spinellis, D. und Szyperski, C. (2004), 'How is Open Source Affecting Software Development?', *IEEE Software* **21**(1), S. 28–33. http://csdl.computer.org/comp/mags/so/2004/01/s1028.pdf [08. Dez 2006].

Torvalds, L. und Diamond, D. (2001), *Just for Fun: The Story of an Accidental Revolutionary*, Harper Business, New York.

Vujovic, S. und Ulhøi, J. P. (2006), An Organizational Perspective on Free and Open Source Software Development, *in* J. Bitzer und P. J. H. Schröder (Hrsg.), 'The Economics of Open Source Software Development', Elsevier Science, Amsterdam, S. 185–206.

Xu, J., Christley, S. und Madey, G. (2006), Application of Social Network Analysis to the Study of Open Source Software, *in* J. Bitzer und P. J. H. Schröder (Hrsg.), 'The Economics of Open Source Software Development', Elsevier Science, Amsterdam, S. 247–269.