media/rep/

Repositorium für die Medienwissenschaft

Robert A. Gehring

FOSS, die Firma und der Markt

2006

https://doi.org/10.25969/mediarep/12499

Veröffentlichungsversion / published version
Sammelbandbeitrag / collection article

Empfohlene Zitierung / Suggested Citation:
Gehring, Robert A.: FOSS, die Firma und der Markt. In: Jeanette Hofmann (Hg.): *Wissen und Eigentum – Geschichte, Recht und Ökonomie stoffloser Güter.* Bonn: Bundeszentrale für politische Bildung 2006, S. 279–297. DOI: https://doi.org/10.25969/mediarep/12499.

Nutzungsbedingungen:
Dieser Text wird unter einer Creative Commons - Namensnennung - Nicht kommerziell - Keine Bearbeitungen 2.0 Lizenz zur Verfügung gestellt. Nähere Auskünfte zu dieser Lizenz finden Sie hier:
https://creativecommons.org/licenses/by-nc-nd/2.0

Terms of use:
This document is made available under a creative commons - Attribution - Non Commercial - No Derivatives 2.0 License. For more information see:
https://creativecommons.org/licenses/by-nc-nd/2.0

DFG Deutsche Forschungsgemeinschaft

Philipps Universität Marburg

Robert A. Gehring

FOSS, die Firma und der Markt

1. Software und Informationsgesellschaft

Das Fundament zur Informationsgesellschaft wird wesentlich durch Softwareentwicklung gelegt. Software ist es, die unsere Datenströme lenkt, die jenen Prozess am Leben erhält, in dem »der menschliche Verstand eine unmittelbare Produktivkraft und nicht nur ein entscheidendes Element im Produktionssystem«[1] darstellt.

Die instantane Kommunikation in globalisierten Märkten, wie wir sie von den Börsentickern im Fernsehen oder den Auktionen bei Ebay kennen, ist ohne Software nicht denkbar. Aber auch die Kommunikation mit unseren Freunden und Verwandten in aller Welt erfolgt mit Unterstützung von Software: Telefongespräche werden im Telefonnetz durch Software vermittelt; die Briefe werden bei der Post von Maschinen sortiert, die von Software gesteuert werden; das Internet, das unsere E-Mails transportiert, läuft mit Software. Kurz gesagt, Software ist unverzichtbarer Teil des sozialen Gebäudes, das wir *Informationsgesellschaft* nennen. Damit kommt der Informationstechnologie im Allgemeinen und der Software im Besonderen eine vergleichbare Rolle zu wie den Sklaven in der antiken Sklavenhaltergesellschaft, Land in der Agrargesellschaft, Rohstoffen und Energie in der modernen Industriegesellschaft: Software ist strategische Ressource und die Kontrolle über ihren Besitz und ihre Verteilung entscheidet über Macht und Reichtum.

Software gewinnt dergestalt eine politische Dimension und die Auseinandersetzungen um große Softwarehersteller – IBM in den 1960er und 1970er Jahren, Microsoft seit den 1990er Jahren – finden quasi automatisch im politischen Raum statt. Es geht darin um mehr als Marktanteile und Wettbewerbsregeln, macht Software die Informationsgesellschaft doch nicht nur möglich, sondern auch verletzlich. Dort, wo Daten Kapital repräsentieren, entstehen neue Abhängigkeiten, wenn diese Daten in Datenformaten gespeichert und verarbeitet werden, die einer Fremdkontrolle unterliegen. Solche Abhängigkeiten können nicht nur teuer werden, wo es um den Einkauf von Lizenzen für entsprechende Software geht. Sie werden aus der Sicht der Datenbesitzer zu einer Existenzfrage, wenn die Verfügbarkeit

der Daten und Programme der eigenen Einflusssphäre weitgehend entzogen ist. Die unternehmerische Freiheit findet dann ihre Grenzen in der Produktpolitik eines Softwareherstellers und nicht mehr nur in der Verfassung.[2] Proprietäre Software[3] mit proprietären Datenformaten liefert ihre Anwender, so gesehen, einer prekären Situation aus, und das Unbehagen darüber ist nur zu verständlich. Das Streben nach Sicherheit leitet die Suche nach Alternativen.

Die ökonomischen Besonderheiten von Netzwerkgütern,[4] die bei Software voll zum Tragen kommen, haben in einem unregulierten Markt zu einer hohen Konzentration geführt. Marktanteile von über 90 Prozent in bestimmten Segmenten sind nicht ungewöhnlich, Marktanteile von mehr als zwei Dritteln üblich. Märkte mit derart hoher Konzentration, man kann in einigen Fällen von der Ausbildung natürlicher Monopole sprechen, lassen die vollständige Konkurrenz des ökonomischen Standardmodells vermissen. Sie neigen dazu, unerwünschte Wettbewerbsergebnisse herbeizuführen.[5] In der Folge werden Ressourcen falsch gelenkt, durch den Anbieter verursachte Kosten nicht durch diesen getragen und Teile der Nachfrage nicht bedient.

In den letzten zwei Jahrzehnten hat sich neben proprietärer Software eine andere Art von Software einen Platz im Softwareuniversum erobert: Freie und Open-Source-Software, kurz FOSS.[6] Diese unterscheidet sich von proprietärer Software in vielerlei Hinsicht, nicht zuletzt in den Institutionen, auf denen sie aufbaut, und den ökonomischen Machtverhältnissen, die sie herbeiführt. Der Aufstieg von FOSS ist eng verknüpft mit dem Aufstieg des Internets. FOSS steuert das Internet zu erheblichen Teilen,[7] FOSS wird in lokalen und globalen Gemeinschaften im Internet verteilt entwickelt und über das Internet verbreitet, ohne dass dafür exklusive Eigentumsrechte nötig wären. Knappheit am Informationsgut ist, abweichend von den üblichen Theorien zum geistigen Eigentum,[8] keine Voraussetzung im FOSS-Modell. Stattdessen herrscht ein Überfluss an Code. Diesem Umstand hat es FOSS zu verdanken, dass dem Phänomen in den letzten Jahren viel Aufmerksamkeit zuteil geworden ist, sowohl von Seiten der wissenschaftlichen Forschung als auch durch die populäre Presse.

Für wohl die meisten akademischen Ökonomen stellt das FOSS-Paradoxon eine echte Herausforderung dar: Wieso funktioniert das FOSS-Modell? Wie kann man mit etwas Geld verdienen, das niemandes exklusives Eigentum ist? Oder etwas allgemeiner gefragt: Wie lassen sich Ressourcen effizient und nachhaltig bereitstellen und bewirtschaften, wenn diese nicht ihrem Wesen nach knapp sind? Müssen öffentliche Güter[9] nicht zwangsläufig der »Tragödie der Allmende«[10]zum Opfer fallen, wie es Garret Hardin vorhergesagt hat? Würde es nicht zu einem Mangel an Software kommen,

wenn diese nicht exklusiv vermarktet werden könnte, wie es die ökonomische Standardtheorie vorhersagt?

Der vorliegende Aufsatz geht diesen Fragen nach und sucht Antworten auf die Frage nach dem Verhältnis von FOSS, Unternehmen und dem Markt. Dabei wird ausgehend von den ökonomischen Randbedingungen diskutiert, auf welchem Fundament FOSS seit über zwei Jahrzehnten wächst und gedeiht. Statt einer »Tragödie der Allmende« (Hardin) erleben wir eine »Komödie der Allmende«.[11]

2. Was ist FOSS?

Die Andersheit von FOSS gegenüber proprietärer Software ruht auf drei Pfeilern:
(1) einer rechtlichen Konstruktion, dem so genannten »Copyleft«,
(2) dem Vertrieb von FOSS im Quellcode und
(3) dem Community-basierten Entwicklungsmodell, das sich vom firmenzentrierten Entwicklungsmodell für proprietäre Software unterscheidet.

FOSS im Recht

Historischer Ausgangspunkt für die gesetzliche Behandlung von Software war ihre Eigenschaft, die darin verkörperten Ideen als Text darzustellen.[12] Abstrakt lässt sich formulieren: *Software ist an eine Maschine gerichtete Information, die textuell repräsentiert ist.* Die Tatsache, dass es sich um Ideen verkörpernde Texte handelt, ist von entscheidender juristischer Bedeutung, denn Texte unterliegen dem Schutz des Urheberrechts:«Zu den geschützten Werken der Literatur, Wissenschaft und Kunst gehören insbesondere: 1. Sprachwerke, wie Schriftwerke, Reden und Computerprogramme.«[13] Den gemeinsamen Rahmen für alle Softwaremodelle bildet also das Urheberrecht,[14] das dem Urheber grundsätzlich die Verfügungsgewalt über die geschaffene Software zuspricht. Der Urheber eines Computerprogramms darf somit allein darüber entscheiden, ob und wie das Programm veröffentlicht, verwertet, vervielfältigt, bearbeitet und verbreitet werden darf. In der Ausübung dieser Rechte zeigt sich dann, welches Modell der Urheber unterstützt, FOSS oder proprietäre Software.

Die Übertragung von urheberrechtlichen Befugnissen erfolgt bei Software entweder auf individueller Vertragsbasis oder durch Standardlizenzen.

Individuelle Verträge kommen üblicherweise bei Individualsoftware, Standardlizenzen bei Software für den Massenmarkt zum Einsatz. Proprietäre Software und FOSS unterscheiden sich lizenzrechtlich in der Hauptsache dahingehend, dass bei FOSS weitergehende Rechte eingeräumt werden. Das Konzept dahinter nennt sich »Copyleft«, in Anlehnung an das anglo-amerikanische Gegenstück zum Urheberrecht, das »Copyright«.

Eine dem Copyleft-Gedanken folgende Lizenz für eine Software gestattet jeder und jedem, die Software zu vervielfältigen, Kopien lizenzkostenfrei weiter zu verbreiten, die Software für beliebige eigene Zwecke zu nutzen und zu bearbeiten. Software unter einer FOSS-Lizenz entbindet die Anwender von aufwendigen Verhandlungen über die Nutzungsrechte an der erworbenen Software. Sinn und Zweck des Copyleft ist es, auf der Basis gemeinsamer Interessen von Entwicklern und Anwendern die Kooperation zwischen ihnen zu fördern sowie die Integration fremder und eigener Leistungen möglichst unaufwendig zu gestalten.

Demgegenüber reservieren Lizenzen für proprietäre Software praktisch alle weitergehenden Rechte – über das der einfachen Nutzung hinaus – für den Eigentümer der Software. Die Integration fremder und eigener Leistungen soll so erschwert werden, um dem Eigentümer ein Alleinstellungsmerkmal im Softwaremarkt zu garantieren, aus dem dieser Kapital schlagen kann.[15]

An dieser Stelle ist auf einen wichtigen Unterschied zwischen Freier und Open-Source-Software hinzuweisen: Bei Freier Software wird das Copyleft »vererbt«, das heißt, jede Version, die in Umlauf gebracht wird, muss ihrerseits mit denselben Befugnissen ausgestattet werden, auch wenn sie bearbeitet wurde. Paradigmatisch ist die GNU General Public License (GPL). Die GPL wird als »viral« bezeichnet, weil jeder aus GPL-Code abgeleitete Code ebenfalls unter der GPL zu lizenzieren ist. GPL-Software »vererbt« in diesem Sinne die an sie gekoppelten Rechte und Pflichten. Bei Open-Source-Software kommen liberalere Lizenzen zum Einsatz, die zum Teil die Privatisierung bearbeiteter Versionen nicht ausschließen.

Quellcode versus Binärcode

Software entsteht als Text. In speziellen Programmiersprachen von Menschen verfasst, werden die Programmtexte – die Quellcodes – von anderen Programmen in die Maschinensprache des jeweiligen Mikroprozessors übersetzt. Erst in dieser Form, praktisch nicht mehr für den Menschen verständlich, werden die Informationen aus den Programmtexten für die Com-

puter interpretierbar. Hierin gleichen sich proprietäre und Freie bzw. Open-Source-Software. Die Besonderheit von solchen Programm-*Texten* liegt darin, dass sie in Verbindung mit entsprechender Hardware Wirkungen[16] auslösen, ohne dass zur Steuerung der Abläufe ein weiteres menschliches Eingreifen notwendig wäre. Computerprogramme ähneln mit diesem Verhalten zwar eher Maschinen als Büchern,[17] entstehen aber eher wie Bücher als wie Maschinen.

Das Recht des geistigen Eigentums kennt eine Dichotomie zwischen ungeschützten, abstrakten Ideen und ihren schutzfähigen Ausdrucksformen (Urheberrecht) bzw. angewandten Wirkmechanismen (Patentrecht[18]). Damit soll einerseits den Interessen der Öffentlichkeit an möglichst weiter Verbreitung der Ideen und andererseits den Interessen der Urheber bzw. Erfinder an der exklusiven Vermarktung der auf den Ideen basierenden Werke bzw. Erfindungen Rechung getragen werden. Der beschriebene Ablauf bei der Softwareentwicklung, wo die im Quelltext konkret ausgedrückten Ideen maschinell in einen nur noch maschinenlesbaren Text übersetzt werden, führt diesen Grundgedanken der freien Verbreitung von ungeschützten Ideen *ad absurdum*. Praktisch niemand ist in der Lage, aus dem Binärcode eines Programms die ihm zu Grunde liegenden Ideen zu extrahieren.[19] Der Zugang zu den ungeschützten Ideen erfordert daher den Zugang zum Quelltext.

FOSS entspricht diesem Bedürfnis nach Zugang und kommt entweder unmittelbar als Quelltext zum Anwender, der den Prozess der Übersetzung dann selbst initiieren muss, um ein ablauffähiges Programm zu erhalten. Oder der Quelltext wird, falls FOSS zur Entlastung der Anwender im Binärcode verbreitet wird, zusätzlich auf Abruf zur Verfügung gestellt. Damit bilden Quelltext und FOSS-Lizenz zwei Seiten einer Medaille, deren Besitz Freiheiten schenkt, die im proprietären Softwaremodell unbekannt sind. Die Nutzung dieser Freiheiten hat zur Herausbildung einer aktiven Gemeinschaft von Softwareentwicklern, -vermarktern und -anwendern geführt, die schlicht als *Community* bezeichnet wird.

Die FOSS-Community und der FOSS-Prozess

Industrielle Produktion von Gütern findet in einem arbeitsteiligen Prozess statt, dessen wesentliche Träger in der Marktwirtschaft Unternehmen – Firmen – sind, die über den Markt miteinander und mit den Konsumenten der Güter im Austausch stehen. Proprietäre Software bildet da keine Ausnahme. Eine Firma ist ein Zusammenschluss von Menschen, die miteinander über-

wiegend in vertraglich fixierten, auf längere Dauer angelegten, hierarchischen Beziehungen verbunden sind. Der Zweck des Zusammenschlusses ist die Produktion und Vermarktung von Waren im weitesten Sinne mit dem Ziel der Erwirtschaftung von Profiten. Um die Profite zu maximieren, versucht die Firma, unnötige Ausgaben zu vermeiden. Die Grenzen der Firma werden maßgeblich dadurch bestimmt, dass die zur Wertschöpfung notwendigen Schritte im Rahmen der Firma kostengünstiger ausgeführt werden können als über den Markt.[20]

Die Entwicklung von FOSS findet, wie erwähnt, in einer *Community* statt,[21] zu der Mitglieder aus den unterschiedlichsten Umfeldern (Mitarbeiter aus Firmen, Forschungseinrichtungen, Universitäten sowie Privatleute) gehören.[22] Die Arbeitsteilung erfolgt dabei im Wesentlichen nicht über Markttransaktionen und Firmenhierarchien, auch wenn Firmen oder Mitarbeiter aus Firmen in der Community aktiv sind und die meisten Projekte in der Community partiell hierarchische Strukturen aufweisen. Nur ausnahmsweise sind die Beziehungen zwischen den Mitarbeitern an einem Projekt vertraglich fixiert, beispielsweise innerhalb von in der Community engagierten Firmen. So kann es vorkommen, dass Unternehmen Mitarbeiter extra für die FOSS-Entwicklung einstellen.

Vorherrschend sind in der FOSS-Community Ad-hoc-Beziehungen aus der Beteiligung von Anwender-Entwicklern, wobei die individuelle Position von der Bedeutung der eigenen Beiträge für das jeweilige Projekt abhängt (Meritokratie). Die Bewertung erfolgt dabei durch die anderen Projektteilnehmer. Dieser Bewertungsprozess wird häufig mit dem aus der Wissenschaft bekannten Peer-Review-Verfahren verglichen, bei dem die Beurteilung der individuellen Leistungen von Wissenschaftlern durch deren Kollegen erfolgt Der offene Peer-Review-Prozess der FOSS-Community findet im Internet statt[23] und ist für Beteiligte und Unbeteiligte weitgehend transparent. Die Projektführung liegt in den Händen charismatischer Persönlichkeiten mit hoher Reputation innerhalb der Community. Deren Autorität beruht auf der Duldung durch die Projektmitglieder, besonders durch die Mitglieder eines engeren Zirkels (»peers«). Solange der oder die Projektführer konform mit den Auffassungen einer Mehrheit von Mitgliedern im Hinblick auf das Projekt sind, behalten sie ihre Rolle. Verstoßen sie jedoch dagegen, laufen sie Gefahr, ihre Position zu verlieren.

Ein FOSS-Prozess (Projekt) kann auf unterschiedliche Weise begründet werden. Um die Endpunkte des Spektrums zu skizzieren: Es kann sein, dass ein individueller Entwickler zur Lösung eines individuellen Problems Software schreibt, deren Quellcode unter einer FOSS-Lizenz im Internet veröffentlicht und zur Mitarbeit einlädt.[24] Oder es kann sein, dass ein Unter-

nehmen ein komplettes Produkt unter einer FOSS-Lizenz auf seinen Webservern zur Verfügung stellt, so dass potentielle Anwender und Entwickler weltweit Zugang zum Quellcode erhalten. Diese nutzen, inspizieren und testen den Code. Ein Teil von ihnen erweitert und verbessert die Software. Die Verbesserungen fließen dann zurück in das Projekt.[25] Mit der Initiierung ist der Erfolg eines FOSS-Projekts noch nicht garantiert, aber eine unabdingbare Voraussetzung geschaffen. Über den Erfolg entscheiden letztlich, wie im Markt, die Abnehmer, d. h. die Anwender. Bedient die Software weit verbreitete Bedürfnisse, steigt die Wahrscheinlichkeit, dass sich um die Software eine Anwender-Entwickler-Community bildet und deren Entwicklung aktiv vorantreibt. Die Entstehung einer Community ist jedoch eher unwahrscheinlich, wenn es sich um eine Spezialsoftware mit sehr engem Anwendungsbereich handelt.

Anwender spielen im FOSS-Modell eine besondere Rolle. Dort, wo es im proprietären Modell Konsumenten gibt, deren Einfluss auf die Produktgestaltung von Massenmarktsoftware eher marginal ist, steht im FOSS-Modell der Anwender-Entwickler, dem die FOSS-Lizenz das Recht und der Quellcode die praktische Möglichkeit geben, eigene Vorstellungen in ein Produkt hineinzuprogrammieren. Über das Internet stehen verschiedene Wege zur Kommunikation mit dem Projekt, d. h. anderen Anwender-Entwicklern offen. Der Anwender-Entwickler ist der *Schlüssel des FOSS-Prozesses*. Statt bloß als Abnehmer aufzutreten, wird er Akteur des Entwicklungsprozesses.

Strukturell handelt es sich bei der FOSS-Community als Ganzem um ein Netzwerk aus Netzwerken (kleineren Communities), wie sie für die Informationsgesellschaft als typisch angesehen werden.[26] Diese Netzwerke verfolgen je eigene Projektziele und produzieren Code, der auf Grund der beschriebenen offenen Lizenzierung von anderen Netzwerken (ganz oder in Teilen) wiederverwendet werden kann. Der Code bildet, so gesehen, den kleinsten gemeinsamen Nenner der Kommunikation zwischen den Netzwerken. Große FOSS-Projekte, wie der Betriebssystemkern *Linux* oder der Webserver *Apache*, differenzieren im Laufe ihrer Evolution Subnetzwerke zur Lösung von Detailproblemen aus. Das können Hardwaretreiber im Fall von Linux oder Module zur Anbindung von Datenbanken im Fall von Apache sein. Das Internet liefert die logistische Grundlage und hat mit seinen niedrigen Kommunikationskosten entscheidend zum Erfolg des FOSS-Modells beigetragen. Die Kommunikation selbst findet zwischen den Mitgliedern der Community *unmittelbar* statt.[27] Das ist ein großer Unterschied zur vermittelten Kommunikation marktwirtschaftlicher Transaktionen, in denen der Preismechanismus als Vehikel dient.[28]

Die Ursprünge der FOSS

An dieser Stelle soll kurz auf die historischen Hintergründe der FOSS-Bewegung eingegangen werden. Will man FOSS nicht bloß als konsequente Übertragung des von der Wissenschaft entwickelten Entdeckungsmodells auf die Software verstehen, lässt sich die erste Hacker-Community im Umfeld des Labors für künstliche Intelligenz am MIT ausmachen. Dort nutzten zu Beginn der 1960er Jahre Studenten, Mitarbeiter und ihre Kinder den liberalen Umgang der Laborleiter, um erste Erfahrungen mit der interaktiven Programmierung von Computern zu sammeln. Zu diesem Zeitpunkt war interaktive Programmierung praktisch unbekannt, es herrschte der so genannte Batch-Betrieb mit Großrechnern vor. Der Zugang zu den Großrechnern war streng hierarchisch organisiert. Software wurde mit Papier und Bleistift entwickelt, dann in Lochkarten gestanzt und den Systemverwaltern übergeben, die den Computer damit fütterten. Der Programmierer selbst kam mit dem Computer kaum in Berührung, durfte sich nur irgendwann die Rechenergebnisse abholen. Gegen dieses Klima der Technikkontrolle rebellierten einige bastelfreudige Individualisten, überwiegend Studenten, und eroberten sich ungenutzte »Kleincomputer«, die sie für ihre Zwecke in Beschlag nahmen. Sie programmierten Betriebssysteme, Assembler und erste Computerspiele, deren Code sie untereinander austauschten und freigiebig an Interessenten außerhalb des Labors weitergaben.

In den 1970er Jahren war Richard Stallman einer der aktiven Hacker dieser Community. Er musste miterleben, wie Kontroll- und Eigentumsdenken immer mehr auch im AI-Labor des MIT um sich griffen. Passwörter wurden eingeführt und Software wurde privatisiert (viele Hacker gingen in die Industrie, gründeten eigene Softwareunternehmen). Die historische Erfahrung des Zusammenbruchs der Hacker-Bewegung lehrte Stallman, dass Softwareentwicklung unter puren Laissez-faire-Bedingungen allzu leicht ein Opfer der »Tragödie der Allmende« werden kann: Alle mit Zugang zum Code können sich diesen aneignen und profitabel verwerten, und überwiegend werden sie genau das tun, wenn sie nicht durch entsprechende institutionelle Rahmenbedingungen daran gehindert werden. Den entnommenen Code werden sie dann als ihr Eigentum betrachten, weiterentwickeln und exklusiv vermarkten.

Software sollte frei sein, war das Motto der Hacker und Stallman glaubte daran. Ende der 1970er Jahr war er praktisch »der letzte der wahren Hacker« (Steven Levy) und beschloss 1983, das MIT zu verlassen: »Aber er verließ das MIT mit einem Plan: eine Version des populären Betriebssystems UNIX zu schreiben und an alle zu verteilen, die daran Interesse hätten.«[29] Er nannte

sein geplantes System GNU, ein Akronym für *GNU is not UNIX*. Schritt für Schritt entwickelte Stallman, unzweifelhaft einer der besten Softwareentwickler der Welt, die zu einem UNIX-System[30] gehörenden Werkzeuge, und fand im Rahmen der von ihm mitbegründeten Free Software Foundation (FSF) Unterstützung durch andere Hacker.

Der Erfolg eines Softwaremodells, das den freien Umgang mit dem Quellcode gestattete, erforderte jedoch die Schaffung von spezifischen institutionellen Rahmenbedingungen, die den Raubbau an der »Code-Allmende« verhinderten. Um zu garantieren, dass die von ihm und seinen Mitstreitern geschriebene Software auch wirklich frei blieb, entwickelte er die erste Lizenz für freie Software, die GNU General Public License (GPL). Anfang der 1990er Jahre waren die Entwicklungsarbeiten weit fortgeschritten, es fehlte praktisch nur noch der Betriebssystemkern. Dessen Platz sollte innerhalb kurzer Zeit Linux ausfüllen, das seit 1991 unter Leitung des finnischen Studenten Linus Torvalds im Internet entwickelt und von diesem unter der GPL lizenziert wurde. Mitte der 1990er Jahre war die Kombination aus GNU und Linux reif genug für erste geschäftskritische Anwendungen und der Apache Webserver machte sie in aller Welt bekannt.

Der Erfolg Freier Software weckte das Interesse von Unternehmern und man begann, vorrangig in den USA, darüber nachzudenken, wie man damit Geld verdienen könnte. Das philosophische Grundkonzept der GPL mit seinem radikalen Freiheitsbekenntnis wurde als zu strikt angesehen und von einer Gruppe von Leuten um Eric S. Raymond modifiziert. Statt von *Freier Software* wollte man zukünftig von *Open-Source-Software* sprechen und eigene Lizenzen entwickeln, die weniger Freiheit auch zulassen würden. Praktisch alle großen Softwareanbieter bieten inzwischen (2005) Teile ihres Produktportfolios für FOSS-Betriebssysteme oder als FOSS an und selbst Microsoft wagt erste Schritte auf die Community zu.[31] FOSS ist für die Wirtschaft attraktiv geworden.

3. FOSS, Firmen und der Markt

FOSS-Prozesse und auch der »Markt« für FOSS entstehen aus der Summe individueller Kosten-Nutzen-Erwägungen und strategischer Überlegungen, die im Einzelnen gar nicht objektiv richtig sein müssen, in ihrer Gesamtheit jedoch ein Wechselspiel von Nachfrage und Angebot hervorbringen, das sich gar nicht so sehr von anderen Märkten unterscheidet. FOSS-Prozesse treten an die Stelle des Marktes, wo diese Organisations- und Produktions-

weise vergleichsweise Kostenvorteile zu bieten hat. Viele Firmen haben das begriffen und bemühen sich um eine eigenständige Open-Source-Strategie, um das Beste aus beiden Welten – Markt und Community – für sich nutzbar zu machen.

Die Nachfrageseite

Auf der Nachfrageseite findet man potentiell jedes Unternehmen, dessen Wertschöpfungskette zumindest in Teilen auf dem Softwareeinsatz beruht. Auch Behörden, Bildungseinrichtungen, Privatanwender usw. zählen zu den Nachfragern. Aus Unternehmenssicht ist die Effizienz der Wertschöpfungskette ausschlaggebend für die Erreichung der Unternehmensziele. »Die Informationstechnik durchdringt die Wertschöpfungskette an jedem Punkt und verändert radikal Wertschöpfungsaktivitäten und zwischen ihnen bestehende Verkettungen. Sie beeinflusst aber auch die Wettbewerbsbreite und die Art und Weise, wie ein Produkt die Wünsche des Konsumenten befriedigt. Diese grundlegenden Effekte erklären, warum die Informationstechnik strategische Bedeutung hat und sich darin von vielen anderen Technologien für kommerzielle Anwendungen unterscheidet.«[32]

Je software-intensiver die Wertschöpfungskette ist, desto stärker fallen die Kostenfaktoren Lizenzierung/Entwicklung und Wartung von Software ins Gewicht. Jedes Softwaremodell, das in der Summe niedrigere Kosten für die Wertschöpfungskette verspricht, wird daher von Firmen auf lange Sicht bevorzugt werden. Das FOSS-Modell leistet das, indem die Entwicklung und Wartung der Software zumindest teilweise außerhalb der Firma stattfindet, ohne dass dafür Lizenz- oder Servicekosten anfallen. So ist es nur allzu verständlich, dass viele Firmen Kosteneinsparungen als eines ihrer wichtigsten Motive für den Einsatz von Open-Source-Software in ihrer Wertschöpfungskette nennen. Weitere wichtige Motive sind u. a.:

- Open-Source-Software unterstützt die Innovation in kleinen Unternehmen.
- Die Beiträge und Unterstützung aus der Community sind hilfreich beim Finden und Beseitigen von Fehlern.
- Open-Source-Software ist zuverlässig und von hoher Qualität.[33]

Effektiv handelt es sich für viele Unternehmen um das Auslagern von Teilen der Wertschöpfungskette, wenn sie auf FOSS zurückgreifen oder selbst FOSS bereitstellen: Ehemals private Kosten für Entwicklung und Wartung (des Unternehmens) werden sozialisiert, wohingegen die Abschöpfung des Profits *aus der gesamten Wertschöpfungskette* überwiegend beim Unternehmen

verbleibt. Eine fundamentale Kapitalismusfeindlichkeit, wie gelegentlich unterstellt, wohnt dem FOSS-Modell keinesfalls inne.[34]

Die Anbieterseite

Auf der Anbieterseite ist potentiell jedes Unternehmen zu finden, das entweder Software oder Produkte, die Software enthalten, vermarktet. Dazu kommen andere Organisationen mit eigener Softwareentwicklung und Privatleute, die im Internet FOSS zur Verfügung stellen. Ein Teil der Nachfrage nach reiner Software wird bereits durch deren Angebote bedient, so dass spezialisierte Softwarehersteller partiell entbehrlich werden.[35]

Reine Software und kombinierte Angebote

Aus der Perspektive von mehr oder weniger reinen Softwareanbietern stellt sich die Lage im Hinblick auf FOSS am schwierigsten dar. Dort ist die Wertschöpfungskette stark auf die Softwareentwicklung und den Softwarevertrieb im Lizenzgeschäft fokussiert, deren Voraussetzung die Knappheit an Code ist. Diese Bedingung ist bei FOSS nicht erfüllt.

Kostengünstige oder lizenzkostenfreie Konkurrenz – nicht nur auch aus der FOSS-Community – gefährdet praktisch die gesamte Wertschöpfungskette solcher Unternehmen. Sollte es ihnen nicht gelingen, durch technologische oder rechtliche[36] Alleinstellungsmerkmale diese Konkurrenz zu verhindern bzw. zu verdrängen, werden sie massiv Umsätze, Marktanteile und Profite einbüßen. Alternativ könnten sie diversifizieren und Einnahmequellen aus anderen Produkten schaffen. Auch die Integration von FOSS-Angeboten mit eigenen, proprietären Erweiterungen stellt eine erfolgversprechende Strategie dar.[37] Die durch Urheber- und Patentrecht gesicherten exklusiven Eigentumsrechte ermöglichen es einem Hersteller proprietärer Software, Profite zu erzielen, indem potentielle Konkurrenten daran gehindert werden, durch einfaches Kopieren ein konkurrenzfähiges Produktes auf den Markt zu bringen, ohne dafür selbst Entwicklungskosten tragen zu müssen. Andernfalls könnte der Plagiator leicht die Preise unterbieten und so das Geschäft des ursprünglichen Anbieters untergraben.

FOSS ermöglicht es aber gerade auch potentiellen Konkurrenten, Software zu kopieren, zu variieren und zu vermarkten. Das steigert auf der einen Seite das Angebot, schmälert aber auf der anderen Seite die potentiellen Profite des eigentlichen Entwicklers. Sinken die Profite dadurch soweit, dass die ursprünglichen Produktionskosten nicht amortisiert

werden können, ist zu erwarten, dass die Softwareproduktion eingestellt wird. FOSS-Anbieter standen hingegen von Anfang an vor der Aufgabe, ohne Lizenzeinnahmen wirtschaften zu müssen. Die Lizenzen der Software, die sie vertrieben, folgten praktisch ausschließlich dem Copyleft-Gedanken, was Lizenzgebühren (nicht aber Gebühren für die Distribution) ausschließt.

Ob für die Entwicklung und Vermarktung von Massenmarktsoftware FOSS eine nachhaltige Grundlage bilden kann, bleibt nach aktuellem Kenntnisstand eine nicht eindeutig zu beantwortende Frage. Als Präzedenzfälle kann man vorrangig die Linux-Distributoren heranziehen, die seit mehreren Jahren im Geschäft sind. Deren Profitmargen bleiben auf der einen Seite (wie zu erwarten) deutlich hinter denen der großen Anbieter proprietärer Software zurück. Auf der anderen Seite ist es ihnen gelungen, durch differenzierte Angebote und insbesondere durch komplementäre Dienstleistungen im Markt zu bestehen. Praktisch alle Linux-Distributoren bieten ihre Produkte mit unterschiedlichem Ausstattungsgrad zu unterschiedlichen Preisen an, und daran anschließend Dienstleistungen wie Auftragsentwicklung, Systeminstallation und -integration sowie Wartung. Für ihren Geschäftserfolg ist der *Mix der Angebote* aus FOSS-Basiskomponenten und individuellen Komplementärleistungen entscheidend. Als reiner Softwareanbieter hat sich keiner von ihnen durchgesetzt.

Während die Distributoren den Schwerpunkt ihrer Aktivitäten auf Produkte legen, agieren große und kleine FOSS-Dienstleister im Markt, deren Leistung in der Anpassung von Standard-FOSS-Paketen an individuelle Anforderungen bestehen. Sie unterscheiden sich nicht wesentlich von Systemintegratoren, die mit proprietärer Software arbeiten. Bezahlt wird die Lösung, nicht die Software, wobei unter einer Lösung in der Regel eine Kombination aus Hardware, Software und Dienstleistung zu verstehen ist.

Auftragsentwicklung

Für kleine Unternehmen mit einem Leistungsportfolio, das individuelle Auftraggeber anspricht, erleichtert FOSS den Marktzutritt und die Innovationsaktivitäten erheblich:

- Eingesparte Lizenz- und Entwicklungskosten können in niedrigere Preise an den Auftraggeber weitergegeben werden, was den potentiellen Markt vergrößert.
- Es gelingt in kürzerer Zeit, Entwicklungskosten zu amortisieren, wenn man auf einem großen vorhandenen Bestand an Software aufbauen kann.

– Ein fehlendes Lizenzgeschäft stellt kein Problem dar, da in der Regel Einnahmen aus Werkverträgen existieren, die reale Kosten abbilden. Die Notwendigkeit, hohe Entwicklungskosten auf zahlenmäßig viele Anwender verteilen zu müssen, entfällt.

Im Fall von Software stimuliert FOSS die Aktivitäten von kleinen und mittleren Unternehmen (KMU) und beseitigt Disparitäten zu großen Herstellern, die über einen eigenen Bestand an Quellcodes verfügen. Konsequenterweise kann man regional ein Wachstum von industriellen FOSS-Strukturen besonders im KMU-Segment beobachten.[38]

Eingebettete Systeme

Eingebettete Systeme sind jene kleinen Computer, die unsere Waschmaschinen, Digitalkameras, Fahrscheinautomaten und Autos steuern. Sie tragen unterstützend zur Gesamtfunktion eines Produktes bei, wir erwerben sie nicht separat.[39] Die Software für eingebettete System ist häufig einer der größten Kostenfaktoren bei der Herstellung komplexer Konsumgüter. Nimmt man als Beispiel die Autoindustrie, so summiert sich der Anteil der das Auto kontrollierenden Software auf bis zu 40 Prozent der Gesamtkosten.[40] Der Drang, durch Verringerung des Entwicklungsaufwandes Kosten zu sparen, ist da nur natürlich.

Eingebettete Systeme kommen in den unterschiedlichsten Zusammenhängen zum Einsatz, entsprechend heterogen sind die Anforderungen an die Software. Ohne Anpassungen lässt sich praktisch kein Code verwenden, sei er proprietärer Natur oder FOSS. Was vor allem zählt, ist Flexibilität. Das betrifft sowohl die Technik als auch die rechtlichen Rahmenbedingungen. FOSS hat in diesem Umfeld schnell Fuß gefasst, wofür ein Gemenge aus unterschiedlichen Motiven verantwortlich ist.[41] Dazu zählen die Verpflichtung aus der GPL, modifizierten Code freizugeben, die Möglichkeit, Kosten durch Auslagerung von Entwicklungs- und Wartungsaktivitäten zu sparen, und das soziale Motiv, als ein guter Mitspieler anerkannt zu werden, um Kooperationspartner in der Community zu finden (Reziprozitätsprinzip[42]).

FOSS in eingebetteten Systemen bringt für den Anbieter den Vorteil mit sich, dass die Softwarekosten im Preis für das Gesamtprodukt berücksichtigt werden können. Physische Produkte lassen sich im Unterschied zu Software nicht kostenfrei reproduzieren. Das Bündeln von Software mit physischen Produkten ermöglicht daher ein profitables Geschäftsmodell, selbst wenn die Software als FOSS freigegeben wird.

4. FOSS: Perspektiven jenseits klassischer Eigentumsverhältnisse

FOSS ist eine Kombination aus Technik, Recht und sozialen Prozessen zur arbeitsteiligen Produktion von Software in der Informationsgesellschaft. Aus der Hackerbewegung der 1960er Jahre entstanden, hat sich mit FOSS ein Modell herausgebildet, das die Bedürfnisse der vernetzten Informationsgesellschaft an kostengünstigem und flexiblem Zugang zu zuverlässiger Software erfüllt. Für Entwicklungs- und Schwellenländer mit einer dynamisch wachsenden IT-Infrastruktur und hohen Lizenzkosten für importierte Software aus den Industrieländern bietet sich das FOSS-Modell sogar als industriepolitisches Förderinstrument an.

Das Gesamtangebot an verfügbarer Software wächst mit FOSS. Der frei verfügbare Quellcode ermöglicht neuen Unternehmen einen kostengünstigen Zutritt zum Markt, was Konzentrationstendenzen entgegenwirkt. Stattdessen wird der Markt anbieterseitig fragmentiert, lokal werden Strukturen gestärkt. Der so verschärfte Wettbewerb zwingt Hersteller proprietärer Softwareprodukte zur Anpassung des Angebots an die Nachfrage, sowohl bei Preisen als auch bei der Leistung. Profitabilität hängt bei FOSS von der geschickten Kombination unterschiedlicher Wertschöpfungsaktivitäten im Geschäftsmodell ab. Dienstleistungen und integrierte Lösungen statt reiner Softwareprodukte tragen den wesentlichen Anteil zum Geschäft bei.

FOSS und Marktwirtschaft sind offensichtlich keine Gegensätze, obwohl die Eigentumsrechte am Code keineswegs den klassischen kapitalistischen Verhältnissen entsprechen. Wie am Beispiel von FOSS sichtbar wird, stärken schwache Eigentumsrechte an Informationsartefakten den Markt für Integrations- und Dienstleistungen mit diesen Artefakten. Eine künstliche Verknappung durch starke Rechte aus geistigem Eigentum ist im Fall von Software keine allgemein notwendige Bedingung für die Schaffung von Anreizen zur Produktion.

In der Informationsgesellschaft wird die Rolle der Community als einer Quelle des gesellschaftlichen Reichtums gestärkt. In der Community werden Informationsgüter in Teilen gemeinschaftlich verwaltet – als Allmenden.[43] Die Wissenschaft stellt das klassische Beispiel dar, FOSS das modernste. Solche Beobachtungen sprechen dafür, dass vergleichbare Ansätze auch in anderen informationsintensiven Märkten erfolgreich sein werden.

Anmerkungen

1 Vgl. Castells (2001), S. 34.
2 »Code is Law«, postuliert Lessig und meint damit, dass Software zu einer eigenen Regulationskraft neben dem Gesetz geworden ist. Vgl. Lessig (1999).
3 Von lat. *proprietas* – Eigentum, Besitz. Gemeint ist Software, auf die exklusive Eigentumsansprüche bestehen und gegebenenfalls durchgesetzt werden.
4 Der Begriff der Netzwerkgüter beschreibt Produkte, die sich durch Komplementarität, Kompatibilität und Standardisierung auszeichnen. Der Erwerb und Einsatz solcher Produkte zieht auf Konsumentenseite Externalitäten (Kosten und Nutzeffekte, die nicht vom Hersteller des Produkts unmittelbar verursacht werden), Kosten beim Technologiewechsel und daher häufig die Bindung von Kunden an bestimmte Technologien und Anbieter (»lock-in«) nach sich. Dem stehen auf Anbieterseite Skaleneffekte bei Produktion und Vermarktung gegenüber. Das klassische Beispiel sind Telefone und das Telefonnetz, bei denen der Nutzen für den Anwender mit der Anzahl der Anwender steigt. Gleichzeitig sinken für den Anbieter (in gewissem Umfang) die durchschnittlichen Kosten mit jedem neuen Anwender. Noch stärker ausgeprägt sind diese *Netzwerkeffekte* bei Software. Vgl. Shy (2001); Shapiro/Varian (1999).
5 Vgl. Schmidt (1999), S. 32–41.
6 Aufgrund ihrer rechtlichen und technischen Gemeinsamkeiten werden Freie und Open-Source-Software im Rahmen dieses Beitrages als *FOSS* zusammengefasst. Eine umfangreichere Analyse würde signifikante Unterschiede besonders in der dahinter stehenden Ideologie erkennen lassen.
7 Das World Wide Web, für viele Menschen ein Synonym für Internet, basiert softwaretechnisch auf Webservern und Webbrowsern. Während bei den Webbrowsern der Löwenanteil an Microsoft mit dem Internet Explorer fällt, dominiert auf der Serverseite der FOSS-Server Apache mit nahezu 70 Prozent »Marktanteil«. Vgl. NetCraft May 2005 Web Server Survey: [http://news.netcraft.com/archives/web_server_survey.html].
8 Vgl. etwa den Beitrag von Klaus Goldhammer in diesem Band sowie Pethig (1997).
9 Ökonomen unterscheiden zwischen privaten Gütern, die auf Märkten gehandelt werden, und öffentlichen Gütern, die auf Märkten nicht (ohne weiteres) gehandelt werden (können). An privaten Gütern bestehen exklusive Eigentumsrechte, an öffentlichen Gütern können solche (aus verschiedenen Gründen) nicht durchgesetzt werden. Während exklusive Eigentumsrechte bei privaten Gütern deren Nutzung durch Nichteigentümer verhindern können, besteht bei öffentlichen Gütern diese Ausschlussmöglichkeit nicht. Typisch ist für öffentliche Güter auch die Nichtrivalität des Konsums, d. h. das Gut kann von mehreren Individuen genutzt werden, ohne dass der individuelle Nutzen geschmälert wird oder dass für neu hinzukommende Nutzer separate Kosten anfallen. Anders formuliert: Öffentliche Güter sind nicht *per se* knapp, nachdem einmal die Kosten für ihre Bereitstellung aufgebracht worden sind. Vgl. Salvatore (2003), S. 611–614; Baden (1998), S. 52.
10 Vgl. Hardin (1968). Hardin benutzte das einer Gemeinde gehörige Weideland (Allmende, engl. *commons*) als Beispiel für eine sog. Common-pool-Ressource. Wenn dieses Weideland von jedem Bauern kostenlos genutzt werden darf, wird das auf Dauer zur Übernutzung führen und das Weideland wird zerstört. Den Grund sieht Hardin darin, dass jedes Individuum ein Interesse an der Nutzung, aber kein Interesse

an der Pflege der Allmende haben kann, solange ihm nicht im Gegenzug exklusive Eigentumsrechte als Belohnung winken.

11 Vgl. Rose (1986).

12 Die USA nahmen als erstes Land 1980 Software als Schutzgegenstand in ihr Copyright-Gesetz auf. Vgl. Merges/Menell/Lemley (2000), S. 911.

13 Vgl. Gesetz über Urheberrechte und verwandte Schutzrechte (Urheberrechtsgesetz), § 2, Abs. 1, Ziff. 1.

14 Weitere Rechtsvorschriften u. a. aus Vertrags-, Marken-, Wettbewerbs- und Patentrecht sind zum Teil gleichzeitig anwendbar.

15 Es ist der Grundgedanke des »geistigen Eigentums« (Urheberrecht und Patentrecht), dem Urheber bzw. Erfinder eine Exklusivposition zu garantieren, aus der heraus die Kosten für Entwicklung und Produktion durch Handel der »Ideenprodukte« im Markt amortisiert werden können. Eine umfangreiche ökonomische Diskussion bieten Landes/Posner (2003).

16 An dieser Stelle verlaufen die Grenzen zu den Gebieten der Technik, die dem Patentschutz zugänglich sind, und in den letzten Jahren hat sich eine heftige Debatte darum entzündet, ob Software Patentschutz erhalten dürfe oder nicht. Vgl. Lutterbeck/Gehring/Horns (2000).

17 Vgl. Samuelson/Davis/Kapor/Reichman (1994).

18 Das Patentrecht schützt *Erfindungen*, wobei der Begriff nicht scharf definiert ist.

19 Das Urheberrecht verschärft dieses Problem noch. So heißt es zwar in § 69a UrhG »Ideen und Grundsätze, die einem Element eines Computerprogramms zugrunde liegen (…) sind nicht geschützt«, aber der Zugang zu diesen Ideen und Grundsätzen, der gegebenenfalls durch Dekompilierung des Binärcodes erfolgen müsste, wird vom Gesetz bis auf Ausnahmen für illegal erklärt; vgl. Nordemann/Vinck (1998).

20 Diese Erklärung liefert Ronald Coase in »The Nature of the Firm«, seinem berühmten Aufsatz von 1937, vgl. Coase (1988).

21 Das Community-Modell ist neben Märkten und Staaten ein weiteres erfolgreiches Modell zur Koordination kooperativer Aktivitäten. Vgl. Bowles/Gintis (1998).

22 Vgl. Robles/Scheider/Tretkowski/Weber (2000).

23 In der Hauptsache über Mailinglisten und Quellcode-Archive.

24 So begann die Linux-Entwicklung. Vgl. Torvalds/Diamond (2001).

25 Vgl. für das Beispiel des Apache-Webservers Lakhani/von Hippel (2003).

26 Vgl. u. a. Castells (2001); Tuomi (2002).

27 Durch elektronische Kommunikation, aber auch auf Community-Treffen. Vgl. u. a. Brucherseifer (2004); Ettrich (2004).

28 Die (partielle) Umgehung dieses Ware-Preis-Mechanismus wird von einigen Autoren als grundlegender Mangel von FOSS gesehen. Vgl. Kooths/Langenfurt/Kalwey (2004). Kritisch dagegen: Pasche/von Engelhardt (2004).

29 Vgl. Levy (1984, S. 427.

30 Zur Geschichte von UNIX vgl. Salus (1995).

31 Vgl. Heise Newsticker (2005).

32 Vgl. Porter/Millar (o. J.), S. 148.

33 Vgl. Bonaccorsi/Rossi (2004) und Dies. (2003).

34 Vgl. auch Himanen (2001), S. 53–57; Heller/Nuss (2004).

35 Vgl. von Hippel (2005)

36 Die intensiven Bemühungen um Softwarepatente müssen in diesem Kontext gesehen werden.

37 Als Beispiele können Apple mit MacOS X und Novell mit der Übernahme des Linux-Distributors SuSE dienen.
38 Vgl. die Aktivitäten der Open-Source-Region Stuttgart: [http://opensource.region-stuttgart.de/] oder die Rolle von FOSS-Unternehmen in der Schweiz: Bern und Basel setzen auf Open-Source-Software (2005).
39 In Zukunft werden Appliances eine größere Rolle spielen. Als »Appliances« bezeichnet man Geräte, deren Kernfunktion durch Hardware und Software implementiert wird. Dazu gehören beispielsweise dedizierte Router und Firewalls. Die Abgrenzung zu eingebetteten Systemen ist unscharf.
40 Vgl. Sullivan/Rüdiger (2004).
41 Vgl. Henkel/Tins (2005).
42 Zur ökonomischen Bedeutung des Reziprozitätsprinzips vgl. Fehr/Gächter.
43 Vgl. Lutterbeck (2005); Foray (2004). Das grundlegende Werk zur Allmende-Wirtschaft ist Ostrom (1999).

Literatur

Baden, John A. (1998): A New Primer for the Management of Common-Pool Resources and Public Goods, in: John A. Baden/Douglas S. Noonan (Hrsg.): Managing the Commons, 2. Aufl. Bloomington-Indianapolis, S. 51–62.

Bern und Basel setzen auf Open-Source-Software (2005), in: Internetausgabe der Neuen Zürcher Zeitung vom 18. Februar 2005, http://nzz.ch/2005/02/18/em/articleCLQCK.html

Bonaccorsi, Andrea/Rossi, Cristina (2003): Why Open Source software can succeed, in: Research Policy 32 (6), S. 1243–1258.

Bonaccorsi, Andrea/Rossi, Cristina (2004): Altruistic individuals, selfish firms? The structure of motivation in Open Source software, in: First Monday 9 (1), http://www.firstmonday.org/issues/issue9_1/bonaccorsi/

Bowles, Samuel/Gintis, Herbert (1998): How communities govern: the structural basis of prosocial norms, in: Avner Ben-Ner/Louis Putterman (Hrsg.): Economics, Values, and Organization, Cambridge u. a., S. 206–230.

Brucherseifer, Eva (2004): Die KDE-Entwicklergemeinde – wer ist das?, in: Robert A. Gehring/Bernd Lutterbeck (Hrsg.), Open Source Jahrbuch 2004, S. 65–81.

Castells, Manuel (2001): Das Informationszeitalter, BD. 1: Der Aufstieg der Netzwerkgesellschaft, Opladen.

Coase, Ronald (1988): The Nature of the Firm, in: Ronald H. Coase, The Firm, the Market and the Law, Chicago-London, S. 33–55.

Ettrich, Matthias (2004): Koordination und Kommunikation in Open-Source-Projekten, in: Robert A. Gehring/Bernd Lutterbeck (Hrsg.), Open Source Jahrbuch, S. 179–192.

Fehr, Ernst/Gächter, Simon (1998: How effective are trust- and reciprocity-based incentives, in: Avner Ben-Ner/Louis Putterman (Hrsg.): Economics, Values, and Organization, Cambridge u. a., S. 337–363.

Foray, Dominique (2004): The Economics of Knowledge, Cambridge/Mass.-London.

Hardin, Garret (1968): The Tragedy of the Commons, in: Science 162, S. 1243–1248.

Heise Newsticker (2005): Microsoft will Brücken zur Open-Source-Gemeinde bauen, Meldung vom 2. Mai 2005, http://www.heise.de/newsticker/meldung/print/59201

Heller, Lydia/Nuss, Sabine (2004): Open Source im Kapitalismus: Gute Idee – falsches System?, in: Robert A. Gehring/Bernd Lutterbeck (Hrsg.), Open Source Jahrbuch, S. 385–405.

Henkel, Joachim/Tins, Mark (2005): Die industrielle Nutzung und Entwicklung von Open-Source-Software: Embedded Systems, in: Bernd Lutterbeck/Robert F. Gehring/Matthias Bärwolff (Hrsg.): Open Source Jahrbuch 2005, Berlin, S. 123–138.

Himanen, Pekka (2001): Die Hacker-Ethik und der Geist des Informations-Zeitalters, München.

Kooths, Stefan/Langenfurt, Markus/Kalwey, Nadine (2004): Open-Source-Software: Eine volkswirtschaftliche Bewertung, MICE Economic Research Studies, Bd. 4, Münster, http://mice.uni-muenster.de/mers/mers4-Open Source_de.pdf

Lakhani, Karim R./von Hippel, Eric (2003): How open source software works: Free user-to-user assistance, in: Research Policy 32 (6), S. 923–948.

Landes, William M./Posner, Richard A. (2003): The Economic Structure of Intellectual Property Law, Cambridge-London.

Lessig, Lawrence (1999): Code and other Laws of Cyberspace, New York.

Levy, Steven (1984): Kackers. Heroes of the Computer Revolution, Gaden City/NY.

Lutterbeck, Bernd (2005): Infrastrukturen der Allmende – Open Source, Innovation und die Zukunft des Internets, in: Bernd Lutterbeck/Robert F. Gehring/Matthias Bärwolff (Hrsg.): Open Source Jahrbuch 2005, Berlin, S. 329–346.

Lutterbeck, Bernd/Gehring, Robert/Horns, Axel H. (2000): Sicherheit in der Informationstechnologie und Patentschutz für Software-Produkte – ein Widerspruch? Kurzgutachten im Auftrag des Bundesministeriums für Wirtschaft und Technologie, Berlin, Dezember 2000, http://ig.cs.tu-berlin.de/forschung/IPR/LutterbeckHornsGehring-KurzgutachtenSoftwarePatente-122000.pdf

Merges, Robert P./Menell, Peter S./Lemley, Mar A. (2000): Intellectual Property in the New Technology Age, 2. Aufl. New York.

Nordemann, Wilhelm/Vinck, Kai (1998): Kommentar zu § 69 d, in:Friedrich Karl Fromm/Wilhelm Nordemann: Urheberrecht: Kommentar, 9. Aufl. Stuttgart, S. 493.

Ostrom, Elinor (1999): Die Verfassung der Allmende: Jenseits von Staat und Markt, Tübingen.

Pasche, Markus/von Engelhardt, Sebastian (2004): Volkswirtschaftliche Aspekte der Open-Source-Softwareentwicklung, Arbeits- und Diskussionspapiere der Wirtschaftswissenschaftlichen Fakultät der Friedrich-Schiller-Universität Jena, Nr. 18.

Pethig, Rüdiger (1997): Information als Wirtschaftsgut in wirtschaftswissenschaftlicher Sicht, in: Herbert Fiedler/Hanns Ulrich (Hrsg.): Information als Wirtschaftsgut: Management und Rechtsgestaltung, Köln, S. 1–28.

Porter, Michael/Millar, Victor E. (o.J.): Wettbewerbsvorteile durch Information, in: Harvard Manager, Informations- und Datentechnik, Bd. 1, Hamburg, S. 146–155.

Robles, Gregorio/Scheider, Hendrik/Tretkowski, Ingo/Weber, Niels (2000): WIDI: Who Is Doing It? A research on Libre Software developers, Forschungsbericht, Fachgebiet Informatik und Gesellschaft der TU Berlin, Berlin, http://ig.cs.tu-berlin.de/lehre/s2001/ir2/ergebnisse/OSE-study.pdf

Rose, Carol (1986): The Comedy of the Commons: Custom, Commerce, and Inherently Public Property, in: University of Chicago Law Review 53, S. 711–781.

Salus, Peter H. (1995): A Quarter Century of Unix, Reading u. a.

Salvatore, Dominick (2003): Microeconomics: Theory and Application, 4. Aufl. New York-Oxford.

Samuelson, Pamela/Davis, Randall/Kapor, Mitchell D./Reichman, J.H. (1984): A Manifesto Concerning the Legal Protection of Computer Programs, in: Columbia Law Review 94 (8), S. 2308–2431, http://www.law.cornell.edu/commentary/intelpro/manif1.htm

Schmidt, Ingo (1999): Wettbewerbspolitik und Kartellrecht, 6. Aufl. Stuttgart.

Shapiro, Carl/Varian, Hal R. (1999):Information Rules. A Strategic Guide to the Network Economy, Boston.

Shy, Oz (2001): The Economics of Network Industries, Cambridge.

Sullivan, Laurie/Rüdiger, Ariane (2004): Autobauer leiden unter Softwarefehlern, in: Internet-Ausgabe der Information Week vom 22. April 2004, http://www.informationweek.de/cms/3051.0.html

Torvalds, Linus/Diamond, David (2001): Just for Fun: Wie ein Freak die Computerwelt revolutionierte, München-Wien.

Tuomi, Ilkka (2002): Networks of Innovation: Change and Meaning in the Age of the Internet, Oxford-New York.

von Hippel, Eric (2005): »Anwender-Innovationsnetzwerke«: Hersteller entbehrlich, in: Bernd Lutterbeck/Robert F. Gehring/Matthias Bärwolff (Hrsg.): Open Source Jahrbuch 2005, Berlin, S. 450–461.