Dieser Artikel ist Teil des

Open Source Jahrbuch 2005

erhältlich unter http://www.opensourcejahrbuch.de.

Das Open Source Jahrbuch 2005 enthält neben vielen weiteren interessanten Artikeln ein Glossar und ein Stichwortverzeichnis.

Ökonomische Eigenschaften von Software – Die Bedeutung von Open-Source-Software für den Wettbewerb auf Softwaremärkten

JENS MUNDHENKE

Dieser Beitrag greift die Beobachtung einer hohen Marktkonzentration auf Märkten für Standardsoftware auf. Es werden die ökonomischen Eigenschaften von Software herangezogen, um diese Marktstruktur und die Dominanz einzelner Hersteller zu erklären und der Frage nachzugehen, inwiefern Open–Source-Software den Wettbewerb auf Softwaremärkten beeinflussen kann. Software als digitales Gut weist besondere Eigenschaften auf: angebotsseitig bestehen Größenvorteile und Verbundvorteile in der Produktion, eine unendliche Skalierbarkeit und eine räumliche Ungebundenheit, nachfrageseitig sind eine Nichtabnutzbarkeit, eine Nichtrivalität im Konsum, Netzwerkeffekte sowie die Eigenschaft eines Erfahrungsgutes gegeben. Zwar ergibt sich damit auch ein Zwang zur steten Innovation und eine hohe Marktdynamik, vor allem aber wird mit den angebots- und nachfrageseitigen wirksamen Größenvorteilen eine Tendenz zur Marktkonzentration impliziert, welche etablierte Anbieter und große Unternehmen bevorzugt. Bei Open-Source-Software werden die Eigenschaften digitaler Güter auf andere Weise als bei proprietärer Software genutzt und andere Marktpräferenzen abgedeckt. Dadurch wird der Wettbewerb direkt (erhöhter Innovations- und Preisdruck) und indirekt (vereinfachter Marktzutritt neuer Konkurrenten) gestärkt. Als öffentliche Ressource mit niedrigen Markteintrittsbarrieren bietet Open-Source-Software zudem ein spezifisches Innovationspotenzial. Abschließend wird auch auf Kritik am Open-Source-Modell eingegangen, die an das Fehlen exklusiver Nutzungsrechte („zu wenig Innovation“) und den mit der Lizenzkostenfreiheit verbundenen Verzicht auf die Preisfunktionen des Marktes („zu schlechte Innovation“) anknüpft.

1. Einleitung

Die Märkte für Standardsoftware sind oftmals durch eine hohe Marktkonzentration und die Dominanz einzelner Anbieter mit Marktanteilen von teilweise mehr als neunzig Prozent gekennzeichnet. Mit der Offenlegung des Quellcodes des Netscape Navigators unter dem Codenamen *Mozilla* sowie der StarOffice-Software unter dem Projektnamen *OpenOffice.org* versuchten Netscape und Sun, einer solche Marktdominanz von Microsoft entgegenzuwirken und durch die Offenlegung ihrer Programme als Open-Source-Software mit Hilfe der Community Marktanteile zurückzugewinnen. Aus der Perspektive der Wirtschafstheorie soll im vorliegenden Beitrag die Frage aufgegriffen werden, welche Innovations- und Wettbewerbspotenziale das Open–Source-Modell bietet.

Anhand der ökonomischen Eigenschaften von Software als digitalem Gut wird zunächst das empirische Bild des Wettbewerbs auf Softwaremärkten, insbesondere die Tendenz zur Marktkonzentration, erklärt. Unter Berücksichtigung der besonderen Eigenschaften von Open-Source-Software in Abgrenzung zu proprietärer Software wird dann diskutiert, wie sich Open-Source-Software auf den Wettbewerb auf Softwaremärkten auswirkt und welche Innovationspotenziale bestehen. Dabei wird auch der Frage nachgegangen, ob das Open-Source-Modell möglicherweise falsche Innovationsanreize setzt.

2. Software als digitales Gut

Software ist ein digitales Gut. Es kann in immaterieller Form, als Folge von Nullen und Einsen dargestellt, im Computer gespeichert und über Datennetze verbreitet werden. Digitale Güter weisen spezifische ökonomische Eigenschaften auf, deren Auswirkungen für den Wettbewerb auf Softwaremärkten im Folgenden beschrieben werden sollen. Dabei werden die Eigenschaften danach unterschieden, ob sie sich auf die Angebots- oder die Nachfrageseite auswirken.[1]

2.1. Angebotsseitige Eigenschaften

Größenvorteile der Produktion Die Produktion von Software als immaterielles Gut basiert auf dem Einsatz von Know-How, das in vollem Umfang bereits bei der Entwicklung der ersten Einheit erforderlich ist. Die Produktionskosten von Software sind somit überwiegend fixe Kosten, während die variablen Kosten der Vervielfältigung, d. h. der Erstellung weiterer Einheiten durch einfaches Kopieren der ersten Einheit, praktisch vernachlässigt werden können. Die Produktion von Software ist somit durch hohe Skalenerträge gekennzeichnet: Je mehr Einheiten produziert werden, desto niedriger sind die durchschnittlichen Kosten.

1 Die folgende Darstellung orientiert sich an Gröhn (1999, S. 4–7), Varian (2000), Katz und Shapiro (1999, S. 31–38), Quah (2003) und Klodt et al. (2003, S. 74–96).

Verbundvorteile der Produktion Die Produktion von Software ist außerdem durch die Wiederverwendung von Programmcode gekennzeichnet. Neue Programmversionen werden auf Basis von bestehendem Code durch Verbesserungen und Ergänzungen entwickelt, und auch bei der Schaffung neuer Software für neue Nutzungszwecke kann auf bestehende Programm-Module und Bibliotheken zurückgegriffen werden. Es bestehen somit Verbundvorteile der Produktion.

Unendliche Skalierbarkeit Digitale Güter lassen sich ohne Qualitätsverlust vervielfältigen. Eine Kopie ist mit dem Original identisch und kann selbst für die verlustfreie Erstellung weiterer Kopien genutzt werden. Wenn man davon absieht, dass notwendige Dienstleistungen für den Vertrieb und den Support von Software nicht kostenfrei skalierbar sind (Liebowitz und Margolis 1999, S. 81 f.), können von einem Softwareprodukt prinzipiell unendlich viele Einheiten produziert werden, es bestehen keine Kapazitätsgrenzen.

Räumliche Ungebundenheit Die Möglichkeit, digitale Güter über Kommunikations- und Datennetze virtuell zu verbreiten, ermöglicht eine räumliche Ungebundenheit der Softwareproduktion. Unabhängig davon, wo ein Programm genutzt wird, kann dessen Produktion (einschließlich der Entwicklung) grundsätzlich überall auf der Welt stattfinden. Es ist daher möglich, dass die Produktion einer Software an einem weltweit einzigen Standort konzentriert wird. Die räumliche Ungebundenheit ermöglicht es aber auch, dass Software in Kooperation von weltweit verteilten Entwicklern erstellt wird, die nur per Internet und ohne direkten persönlichen Kontakt miteinander kommunizieren.

2.2. Nachfrageseitige Eigenschaften

Nichtabnutzbarkeit Software kann als langlebiges Gebrauchsgut prinzipiell unbegrenzt lange genutzt werden. Sie verschleißt nicht, und auch ein intensiver Gebrauch führt nicht zur physikalischen Abnutzung. Software kann aber veralten: Veränderte inhaltliche Anforderungen, höhere Ansprüche an neue Features sowie die Verfügbarkeit neuer technischer Komponenten (Hardware) machen es erforderlich, neue und verbesserte Programmversionen einzusetzen.

Nichtrivalität im Konsum Die Möglichkeit der leichten und schnellen Vervielfältigung von Software erlaubt es, diese gleichzeitig weiterzugeben und zu behalten. Man kann dies als Nichtrivalität im Konsum interpretieren, da die Nutzung eines Programms durch eine Person andere Personen in der Nutzung des gleichen Programms nicht einschränkt und ein Programm grundsätzlich auf beliebig vielen Rechnern gleichzeitig installiert werden kann.

Nachfrageseitige Größenvorteile durch Netzwerkeffekte Netzwerkeffekte liegen dann vor, wenn der Nutzen eines Gutes für einen einzelnen Anwender umso höher ist, je häufiger sich andere für die Nutzung des gleichen Gutes entscheiden. Netzwerkeffekte können daher als nachfrageseitige Größenvorteile betrachtet werden.

Dabei unterscheidet man im Allgemeinen zwischen direkten und indirekten Netzwerkeffekten. Bei Software können prinzipiell beide Effekte auftreten.

Direkte Netzwerkeffekte beziehen sich auf die als Nutzen stiftend empfundene Möglichkeit, sich innerhalb eines Netzwerks mit anderen auszutauschen. Klassische Beispiele sind Telefon und Faxgerät, da die Möglichkeit, mit anderen zu kommunizieren, um so größer ist, je weiter verbreitet diese Geräte sind. Direkte Netzwerkeffekte sind bei Software etwa durch den Austausch von Daten und Dokumenten von Bedeutung: Die Zusammenarbeit mit anderen an einem gemeinsamen Textdokument erfordert in der Regel das gleiche Textverarbeitungsprogramm. Die Wahrscheinlichkeit, mit einem potenziellen Kooperationspartner zusammenarbeiten zu können, steigt mit der Verbreitung der genutzten Anwendungssoftware.

Indirekte Netzwerkeffekte sind dadurch gekennzeichnet, dass mit zunehmender Verbreitung eines Gutes die Verfügbarkeit an komplementären Gütern und Dienstleistungen zunimmt. Dies erhöht einerseits die Auswahlmöglichkeiten der Nutzer und erhöht anderseits die Attraktivität für die Anbieter komplementärer Produkte, diese für eine große Nutzerbasis verfügbar zu machen. Indirekte Netzwerkeffekte sind im Bereich Software insbesondere bei der Entscheidung für ein Betriebssystem von Bedeutung, die eben dadurch beeinflusst wird, wie viele Anwendungsprogramme für ein jeweiliges System verfügbar sind und wie viele *Experten*, die dieses System ebenfalls nutzen, bei Fragen und Problemen Hilfestellung leisten können.

Software als Erfahrungsgut Anders als bei reinen Suchgütern, deren Eigenschaften vorab bekannt sind und die entsprechend zielgerichtet gesucht und ausgewählt werden können, weist Software die (zusätzlichen) Kennzeichen eines Erfahrungsgutes auf, dessen Eigenschaften sich erst im Gebrauch vollständig offenbaren. Beispielsweise lässt sich erst nach dem Kauf einer Software bei regelmäßiger Nutzung feststellen, ob das Zusammenspiel mit den vorhandenen Hard- und Softwarekomponenten fehlerfrei abläuft und alle vorab festgelegten Anforderungen erfüllt werden. Die durch die Benutzung gesammelten Erfahrungen mit einer Software kumulieren zu Lerneffekten, d. h. es bildet sich ein spezifisches Wissen über die Nutzung einer bestimmten Software.

2.3. Implikationen für den Wettbewerb auf Softwaremärkten

Ausschließbarkeit als Voraussetzung für das Entstehen von Softwaremärkten

Die Nichtrivalität im Konsum und die gleichzeitige Möglichkeit der unendlichen Skalierbarkeit der Produktion machen es bei digitalen Gütern schwierig, zahlungsunwillige Konsumenten von deren Nutzung auszuschließen. Die Ausschließbarkeit von Nutzern ist jedoch Voraussetzung dafür, dass sich Märkte herausbilden können und eine private Bereitstellung möglich wird. Andernfalls hätten Anbieter in einem solchen Markt den klassischen Gesetzen der Ökonomie entsprechend aufgrund eines möglichen *Free-Ridings* zahlungsunwilliger Nutzer nur einen verminderten Anreiz, in die

Bereitstellung digitaler Güter zu investieren. Genau wie bei den so genannten *öffentlichen Gütern* bestünde das Problem der Unterversorgung und es wäre sogar denkbar, dass letztlich gar keine funktionsfähigen Märkte entstehen.

Eine hinreichende Ausschließbarkeit von Konsumenten wird bei Software auf rechtlichem Wege durch Nutzungsrechte (Gesetze und Lizenzen) und auf technischem Wege (Kopierschutz, Registrierung, digitale Signaturen) erreicht. Für die weitere Analyse sei unterstellt, dass eine Ausschließbarkeit von Konsumenten realisierbar ist, so dass Märkte für Software existieren. Im Folgenden wird diskutiert, wie die spezifischen Eigenschaften von Software die Struktur und den Wettbewerb eines solchen Marktes beeinflussen.

Tendenz zur Marktkonzentration und zur Bevorzugung etablierter Anbieter

Die Märkte für Standardsoftware sind überwiegend durch eine hohe Marktkonzentration gekennzeichnet, einzelne Anbieter ziehen einen Großteil der Marktnachfrage auf sich (Gröhn 1999, S. 110). Diese Tendenz zur Marktkonzentration und zur Bevorzugung etablierter Anbieter lässt sich auf die Kombination von angebots- und nachfrageseitigen Größenvorteilen bei Software zurückführen.

Auf der Nachfrageseite sorgen Netzwerkeffekte dafür, dass sich die Anwender auf wenige oder sogar nur einzelne Computerprogramme konzentrieren, wenn die gemeinsame Nutzung der gleichen zugrunde liegenden Technologie vorteilhafter ist als die Nutzung eines auf einer anderen Technologie basierenden separaten Produkts ohne ein solches Nutzernetzwerk. Auf der Angebotsseite verstärken die Skaleneffekte und die Verbundvorteile in der Produktion von Software die Konzentration auf wenige Anbieter, die überdies von einem einzelnen Standort aus praktisch die ganze Welt versorgen können.

Eine solche Tendenz zur Dominanz eines Anbieters oder weniger Anbieter ist nicht per se negativ zu beurteilen, da aufgrund des hohen Fixkostenanteils wenige Anbieter den gesamten nachgefragten Output kostengünstiger erstellen können als eine Vielzahl kleiner Anbieter. Für die Anwender ergeben sich damit Preisvorteile. Eine Konstellation, in der sogar nur ein Anbieter die gesamte nachgefragte Menge am effizientesten bereitstellen kann, wird daher auch als *natürliches Monopol* bezeichnet. Eine Marktstruktur mit nur wenigen Anbietern kann zudem auch unter dem Aspekt der Netzwerkeffekte vorteilhaft sein. Sind die Netzwerkeffekte stark genug, entscheiden sich alle Konsumenten für dieselbe Technologie, und es wäre gegebenenfalls nicht sinnvoll, eine Marktstruktur mit vielen kleinen Firmen zu erzwingen (Sachverständigenrat zur Begutachtung der Gesamtwirtschaftlichen Entwicklung 2000, Liebowitz und Margolis 1999).

Netzwerkeffekte als Größenvorteile auf der Nachfrageseite können aber auch anbieterübergreifend auftreten, wenn verschiedene Anbieter einen einheitlichen Standard nutzen. Standards können entweder durch eine direkte Kooperation von Anbietern festgesetzt werden oder sich im Wettbewerb herausbilden. Eine kooperative Standardsetzung ermöglicht einen frühen Wettbewerb innerhalb eines Netzwerkes, kann aber auch dazu führen, dass eine inferiore Technologie ausgewählt wird. Eine kom-

petitive Standardfindung bedeutet, dass *ex ante* keine Festlegung stattfindet und sich ein einheitlicher Standard erst im Wettbewerb zwischen verschiedenen Netzwerken herausbildet. Dabei sind multiple Gleichgewichte möglich, und es ist vorab nicht prognostizierbar, welche Technologie sich dauerhaft durchsetzen wird. An die Stelle des Wettbewerbs im Markt tritt der Wettbewerb um den Markt (Sachverständigenrat zur Begutachtung der Gesamtwirtschaftlichen Entwicklung 2000).

Die Wahl eines Standards entscheidet über das Ausmaß an Kompatibilität zur Technologie eines konkurrierenden Anbieters. Sie stellt damit eine bedeutende strategische Handlungsoption eines Unternehmens dar und bestimmt die Größe eines Netzwerks (Liebowitz und Margolis 1999, S. 88; Mendys-Kamphorst 2002, S. 23). Kleine Unternehmen profitieren verhältnismäßig stärker von der gemeinsamen Nutzung eines Netzwerks und streben daher in der Regel eine höhere Kompatibilität an als ein großer Anbieter. Je größer das eigene Netzwerk – die so genannte *installierte Basis* – desto geringer die Vorteile einer erweiterten Kompatibilität. Bei Marktbeherrschung kann daraus eine aus Anwendersicht zu geringe Kompatibilität resultieren, d. h., dass eine Zusammenarbeit mit den Nutzern einer alternativen, inkompatiblen Technologie erschwert wird.

Problematisch ist eine solche Marktmonopolisierung dann, wenn diesem Monopol ein inkompatibler, technisch unterlegener Standard zugrunde liegt, so dass es aufgrund von Netzwerkeffekten zu Ineffizienzen in der Adoption einer Technologie kommen kann, weil wegen der Abhängigkeit von einer installierten Basis der Wechsel zu einer neuen Technologie mit höheren Kosten verbunden ist als bei normalen Gütern. Im Extremfall besteht die Gefahr eines Lock-in, d. h. des wohlfahrtstheoretisch schädlichen Verharrens in einer veralteten Technologie, obwohl ein kollektiver Wechsel in ein neues Netzwerk alle Nutzer besser stellen würde (Sailer 2001, S. 354–357).

Da der persönliche Nutzen für den Einzelnen unmittelbar auch von der Entscheidung potenzieller und tatsächlicher Mitkonsumenten abhängig ist, sind Erwartungen an das Verhalten anderer für das Marktergebnis, d. h. die Adoption und die Diffusion einer Technologie, von großer Bedeutung. Damit eröffnen sich den Anbietern Spielräume für strategisches Verhalten, um die Erwartungen zugunsten des eigenen Netzwerks zu beeinflussen: Eine Vielzahl strategischer Instrumente, z. B., eine Produkt- und Preisdifferenzierung, die Bündelung von Produkten oder auch die Tolerierung von Raubkopien (Ben-Shahar und Jacob 2001), wird dazu genutzt, potenzielle Nutzer von den Vorteilen einer Technologie zu überzeugen und das empfundene Risiko, sich für die *falsche* Technologie zu entscheiden, zu reduzieren. Unternehmen mit Marktmacht und Reputation sind hierbei gegenüber kleineren Anbietern im Vorteil, da sie aufgrund ihrer Finanzkraft und ihrer Reputation mit höherer Glaubwürdigkeit eine neue Technologie fördern können und auch über monopolistische Gestaltungsspielräume verfügen (vgl. Sailer 2001, S. 352–354; Shapiro und Varian 1999, Kap. 3).

Netzwerkeffekte können somit als Marktbarriere interpretiert werden, welche es Konkurrenten erschwert, in den Markt einzutreten. Eine fehlende Kompatibilität zum herrschenden Standard macht es schwierig, eine Technologie parallel zu einem bestehenden Netzwerk zu etablieren. Lerneffekte und Wechselkosten bewirken, dass sich bestehende Marktstrukturen verfestigen, da etablierte Anbieter über ein breiteres

Spektrum strategischer Instrumente verfügen.

Hohe Marktdynamik durch den Zwang zu Innovation

Wie bislang gezeigt wurde, wirken einige spezifische Eigenschaften von digitalen Gütern und damit auch von Software auf eine Dominanz weniger Anbieter hin. Diese Strukturen können sich weiter verfestigen, und es bestehen Anreize, Marktmacht zu missbrauchen und den Wettbewerb zu beschränken.

Die Existenz von Netzwerkeffekten muss jedoch nicht automatisch zu einer Monopolisierung und einer Zementierung dominanter Marktstrukturen führen. Aufgrund der Eigenschaft von Software als dauerhaftem Gut, das keiner physikalischen Abnutzung unterliegt und theoretisch unbegrenzt lange genutzt werden kann, würde der Nutzer eines Programms dem Anbieter dieser Software im Prinzip unmittelbar nach dem Kauf als Kunde wieder verloren gehen. Um dauerhafte Umsätze erzielen zu können, ist daher auch ein einzelner dominanter Anbieter gezwungen, das von ihm angebotene Produkt kontinuierlich weiterzuentwickeln, es zu verbessern und im Funktionsumfang zu erweitern. Jeder Anbieter konkurriert nicht nur mit seinen direkten Wettbewerbern, sondern auch mit der eigenen installierten Basis.

Die stetige Verbesserung und Erweiterung von Software bewirkt einerseits zwar, dass viele Programme heute eine für den durchschnittlichen Anwender nahezu unüberschaubare Vielzahl an Funktionen aufweisen, von denen meist nur ein kleiner Teil tatsächlich benötigt wird, andererseits trägt der Zwang zum technischen Fortschritt jedoch zu einer hohen Innovationsdynamik der Märkte der Informations- und Kommunikationstechnologien bei. Die Dynamik des Wandels wurde durch die Verbreitung des Internets in den letzten Jahren weiter verstärkt: Märkte entstehen und verschwinden in höherem Tempo als bisher, Produktzyklen werden kürzer, neue Produkte lösen einander rascher ab. Eine schnelle Markteinführung neuer Produkte bietet zudem die Chance, gänzlich neue Märkte zu schaffen und zumindest zeitweise als Monopolist agieren zu können.[2] Gerade in technisch anspruchsvollen Märkten bestehen somit auch für einen dominanten Anbieter starke Anreize, sich wettbewerbsorientiert zu verhalten, denn zumindest in den frühen Marktphasen ist die Marktführerschaft instabil und angreifbar.

Liebowitz und Margolis (1999) können für die achtziger und frühen neunziger Jahre die Fragilität der Marktführerschaft in verschiedenen Marktsegmenten für Standardsoftware empirisch untermauern; langfristig erscheint der technische Fortschritt jedem monopolistischen Beharrungsvermögen überlegen. Fraglich ist jedoch, ob sich die Märkte für Standard-Anwendungssoftware heute nicht bereits in einer reifen Marktphase befinden, in der sich die Marktstrukturen soweit gefestigt haben, dass ein Wechsel der Marktführerschaft weniger wahrscheinlich geworden ist. In allen von Liebowitz und Margolis untersuchten Marktsegmenten (u. a. Textverarbeitung, Tabellenkalkulation, Internetbrowser, PC-Betriebssysteme) ist zu beobachten, dass Microsoft eine einmal eroberte Führungsposition seither nicht mehr abgegeben hat.

2 Man nennt dies auch *First Mover Advantage*, vgl. dazu Monopolkommission (2003) und Sachverständigenrat zur Begutachtung der Gesamtwirtschaftlichen Entwicklung (2000).

3. Wettbewerbs- und Innovationspotenziale von Open Source

3.1. Kennzeichen von Open-Source-Software und proprietärer Software

Die grundlegenden Merkmale von Open-Source-Software sind die Verfügbarkeit des Quellcodes *und* die freie Nutzungslizenz, welche eine uneingeschränkte Nutzung, Ergänzung und Veränderung eines Open-Source-Programms erlaubt und die lizenzkostenfreie Vervielfältigung und Verbreitung zulässt. Die Offenheit des Quellcodes allein ist kein hinreichendes Kriterium für Open-Source-Software, erst mit der gleichzeitigen Einräumung umfassender Nutzungsrechte, die die freie Nutzung, Modifizierung und Weitergabe erlauben, sind die Anforderungen des Open-Source-Prinzips erfüllt.

Im Gegensatz dazu wird proprietäre Software in der Regel nur in Binärform vertrieben und muss durch Kauf *erworben* werden. Der Nutzer erhält lediglich eine einfache Nutzungslizenz, welche das Kopieren und Weitergeben verbietet und somit die Einnahme weiterer Lizenzgebühren sicherstellt. Alle über die einfache Nutzung hinausgehenden Rechte verbleiben beim Softwareproduzenten. Bei proprietärer Software steht somit der Schutz eines „geistigen Eigentums" mit Ausschließlichkeitsrechten besonders im Vordergrund, eine Nutzung wird nur in eingeschränktem Umfang lizenziert.

Neben der freiheitlichen Nutzung unterscheidet sich Open-Source-Software aber auch in seiner Entwicklungsmethode von proprietärer Software. Proprietäre Software wird üblicherweise in einem kontrollierten, streng überwachten Entwicklungsprozess innerhalb eines einzelnen Unternehmens von einem fest umgrenzten Entwicklerteam geschrieben. Neue Versionen werden in verhältnismäßig langen Zeitabständen auf den Markt gebracht und erst nach ausführlichen Tests zum Verkauf freigegeben. Für die Entwicklung von Open-Source-Software hat Raymond (1998) die Metapher eines orientalischen Basars geprägt. Die Open-Source-Entwicklung erfolgt innerhalb einer prinzipiell unbegrenzt großen, weltweiten Community, die sich selbst koordiniert und deren Mitglieder über das Internet miteinander kommunizieren. Jeder Interessierte kann sich beteiligen. Programmierer und Nutzer wirken auf freiwilliger Basis, vielfach unentgeltlich und in ihrer Freizeit, an der Entwicklung neuer und dem Testen bestehender Open-Source-Module mit, so dass eine parallele Weiterentwicklung und Fehlerbeseitigung möglich ist, die im Idealfall deutlich dynamischer verläuft als bei proprietärer Software. Neuere Untersuchungen belegen aber auch einen zunehmenden Einfluss kommerzieller Unternehmen auf die Entwicklung von Open-Source-Software (Ghosh et al. 2002, S. 12 f.).

Wie bereits beschrieben, ist es bei digitalen Gütern theoretisch erforderlich, das Problem der Nichtausschließbarkeit zu lösen, um die Voraussetzungen für eine private Bereitstellung zu schaffen. Bei proprietärer Software wird dies durch technische (Kompilierung in Binärcode, Verschlüsselungsmechanismen) und rechtliche (Schutz durch Urheberrecht und Patente) Konstrukte erreicht, um die Umsetzung von Ideen in fertige Produkte exklusiv vermarkten zu können.

Bei Open-Source-Software werden diese technischen und rechtlichen Konstrukte sowie die Eigenschaften digitaler Güter ebenfalls zielgerichtet genutzt, allerdings zu anderen Zwecken als bei proprietärer Software, nämlich zur bewussten Erhaltung und

Absicherung der Nichtausschließbarkeit sowie zur Dezentralisierung der Entwicklung (Quah 2003, S. 315).

Die Einräumung weitreichender Nutzungsrechte bei Open-Source-Software bedeutet nämlich nicht den Verzicht auf die urheberrechtlich geschützten Verwertungsrechte, sondern das Urheberrecht wird vielmehr bewusst genutzt, um die Verbreitung von Open-Source-Software zu steuern und den Nutzern gewisse Pflichten aufzuerlegen (Jaeger und Metzger 2002, S. 31). Die bedeutendste Klausel dieser Art ist die *Copyleft*-Klausel, welche sicherstellt, dass Open-Source-Software nicht wieder unfrei werden kann. Die *Copyleft*-Klausel ist in einer Reihe von Open-Source-Lizenzen enthalten ist, darunter in der *GNU General Public License* (GPL), der meistgenutzten Open-Source-Lizenz überhaupt.[3]

Die freie Zugänglichkeit des Quellcodes ermöglicht und vereinfacht es, ein Programm an eigene Bedürfnisse anpassen zu können. Änderungen können für andere sofort verfügbar gemacht und von diesen ebenfalls benutzt, auf ihre Eignung und Fehlerfreiheit überprüft und wiederum ergänzt und verbessert werden. Solche Verbesserungen werden dann ihrerseits wieder anderen verfügbar gemacht, so dass sie von anderen gleich eingesetzt werden können.

Die Offenheit und die Verständlichkeit des Quellcodes vereinfachen außerdem die Wiederverwendung von erstelltem Code und die Rekombination von Programmmodulen. Die Nichtrivalität im Konsum wird also bewusst dazu genutzt, einer Vielzahl von Anwendern und Entwicklern gleichzeitig die Fehlerprüfung des Codes und das Testen zu ermöglichen. Durch die Parallelisierung verschiedener Entwicklungsstufen kann der Prozess der Softwareentwicklung beschleunigt werden.

Die räumliche Ungebundenheit digitaler Güter wird schließlich nicht zur räumlichen Konzentration der Softwareerstellung an einem einzelnen Ort genutzt, sondern erlaubt es Entwicklern in aller Welt, an der Erstellung von Open-Source-Software zu partizipieren.

3.2. Wettbewerbswirkungen

Die zunehmende Verbreitung von Open-Source-Software und dem Open-Source–Entwicklungsmodell übt einen positiven Einfluss auf den Wettbewerb in Softwaremärkten aus. Dabei kann zwischen direkten und indirekten Wettbewerbswirkungen unterschieden werden.

Eine direkte Wettbewerbswirkung von Open-Source-Software besteht darin, dass sich das Angebot an Software vergrößert und damit mehr Wahlmöglichkeiten für die Anwender bestehen. Zum einen werden dadurch, dass bei Open-Source-Software die spezifischen Eigenschaften digitaler Güter auf andere Weise genutzt werden als bei herkömmlicher proprietärer Software (Quah 2003, S. 315), andere Präferenzen abgedeckt, d. h., dass die Möglichkeit, den Code anschauen, verändern und ergänzen zu können oder das Programm an andere weitergeben zu dürfen, für bestimmte Anwender selbst Nutzen bringend wirkt. Sofern Open-Source-Programme als qualitativ hochwertige

3 Vgl. http://sourceforge.net/, nach eigener Darstellung „the world's largest Open Source development website". 70 % aller auf Sourceforge gelisteten Open-Source-Projekte werden unter der GPL lizenziert.

und technisch geeignete Alternative, d. h. als Substitut für das bestehende Angebot, wahrgenommen werden, erhöht sich außerdem der Wettbewerbsdruck für die etablierten Anbieter. Die lizenzkostenfreie Verfügbarkeit von Open-Source-Software bewirkt einen erhöhten Preisdruck und es ist zu erwarten, dass sich Softwarepreise reduzieren (Rosenberg 2000, Kap. 12).

Mehr Wettbewerb bedeutet auch mehr Innovationsdruck, d. h. für die Softwareproduzenten bestehen verstärkte Anreize, sich durch Verbesserung der Produktqualität und Investitionen in neue Produkte vom Angebot der Konkurrenz abzugrenzen. Ebenso sind eine stärkere Produktdifferenzierung und eine Berücksichtigung von Kundenbedürfnissen auch in kleineren Marktnischen zu erwarten. Im Sinne von Hayeks (1968) stärkt Open-Source-Software somit den „Wettbewerb als Entdeckungsverfahren" für neue Produkte und Verfahren.

Eine indirekte Wettbewerbswirkung ergibt sich dadurch, dass der potenzielle Wettbewerb im Sinne des Konzepts bestreitbarer Märkte (Baumol 1982, Baumol et al. 1988) erhöht wird. Im Mittelpunkt dieses Konzepts steht die Überlegung, dass nicht eine bestimmte Marktform, sondern das Ausmaß an potenzieller Konkurrenz über das Wettbewerbsverhalten auf einem Markt entscheidet. Sofern ein Markt bestreitbar ist, d. h. wenn ein freier Marktzutritt möglich ist, zwingt die potenzielle Konkurrenz, unabhängig davon, ob ein solcher Markteintritt tatsächlich stattfindet, alle Marktteilnehmer zu wettbewerblichem Verhalten. Es spricht einiges dafür, Softwaremärkte als prinzipiell bestreitbar zu betrachten. Die speziellen Eigenschaften von Open-Source–Software tragen dazu bei, diese Bestreitbarkeit zu erhöhen und mehr potenziellen Wettbewerb zu ermöglichen, da im Vergleich zu proprietärer Software ein besonderes Potenzial für Innovationen besteht, das im nächsten Abschnitt beschrieben werden soll.

3.3. Innovationspotenziale von Open-Source-Software

Die Analyse des Innovationspotenzials von Open-Source-Software knüpft ebenfalls an wesentliche Merkmale des Open-Source-Modells, die Offenheit des Quellcodes und eine freie Lizenz, die eine fast uneingeschränkte Nutzung der Software zulassen, an. Dies eröffnet im Vergleich zu proprietärer Software ein besonderes Potenzial für Innovationen.

Niedrige Markteintrittsbarrieren

Open-Source-Software kann als öffentlich verfügbare Ressource mit niedrigen Markteintrittsbarrieren interpretiert werden, d. h., dass bei Open-Source-Software in besonders effizienter Weise von einem bestehenden Wissenskapital Gebrauch gemacht wird. Die Nachahmung oder sogar die direkte Übernahme technischer Lösungen wird deutlich vereinfacht. Da ein Ausschluss anderer nicht möglich ist und interne Programmroutinen sowie die Spezifikation von Standards nicht geheimgehalten werden können, fällt es leichter, komplementäre Produkte zu entwickeln und anzubieten. Damit wird es insbesondere für kleine Unternehmen einfacher, in einen Markt neu einzutreten. Somit wird die Macht etablierter (und großer) Anbieter reduziert.

Die Nutzung bestehenden Quellcodes als technische Basis, um darauf aufbauend neue Produkte zu entwickeln, bietet verschiedene Ansatzpunkte für die Verwirklichung von Innovation. Zunächst erlaubt die Verfügbarkeit des Codes eine Vermeidung von doppeltem Aufwand, so dass die Entwicklung neuer Produkte beschleunigt wird. Durch die Einsparung von Ressourcen werden Kosten gesenkt. Dieser finanzielle Aspekt wird durch den lizenzkostenfreien Zugang zum bestehenden Code verstärkt. Auch kleine Unternehmen haben somit die Möglichkeit, unter Umgehung zumindest eines Teils der ansonsten anfallenden fixen Kosten neue Produkte auf Basis von existierender Open-Source-Software zu entwickeln und anzubieten. Open-Source–Software unterstützt somit das Prinzip der Wiederverwendung von Code. Ein Beispiel stellt die Entwicklung des Betriebssystems MacOSX durch Apple dar, bei dem auf eine Variante der bestehenden *Berkeley Software Distribution* (BSD, ein Derivat des Unix-Betriebssystems) zurückgegriffen wurde. Durch die Nutzung eines bereits existierenden Betriebssystems als technische Basis konnte die Entwicklungszeit bis zur Markteinführung im Vergleich zu einer kompletten Neuentwicklung erheblich reduziert werden.[4]

Anders als bei proprietärer Software steht die Möglichkeit der Wiederverwendung von Code grundsätzlich allen offen, sofern die Bedingungen der Nutzungslizenz eingehalten werden. Infolgedessen ist die Zahl potenzieller Innovatoren prinzipiell unbegrenzt, und das Innovationspotenzial kann durch einen größeren Anbieterkreis erschlossen werden.

Dadurch, dass Dritte die rechtliche und technische Möglichkeit haben, den Programmcode einer Software einsehen, verändern und verbessern zu können, reduziert sich zudem für die Nutzer von Open-Source-Technologie die Abhängigkeit vom Anbieter dieser Software. Obwohl eine größere Zahl von Anbietern in einem Markt mit einer größeren Unsicherheit darüber einhergeht, welche Unternehmen sich langfristig am Markt behaupten können, besteht eine verminderte Gefahr eines anbieterspezifischen Lock-in, denn wenn der originäre Anbieter einer verwendeten Software den Support und die Weiterentwicklung einstellen würde, könnte diese Dienstleistung auch von anderen Anbietern erbracht werden. Damit reduziert sich die Gefahr, in einer Technologie zu *stranden*.

Die Offenheit des Quellcodes und das Recht, diesen verändern und weiterentwickeln zu dürfen, bietet außerdem das Potenzial einer nutzerinduzierten Innovation. Anders als bei proprietärer Software, die sich zwar auch an den Bedürfnissen der Nutzer orientiert, aber nur vom Anbieter selbst verändert werden kann und darf, haben die Nutzer von Open-Source-Software – Programmierkenntnisse vorausgesetzt – selbst die Möglichkeit, erforderliche Anpassungen durchzuführen und neue Funktionalitäten hinzuzufügen.

Da keine großen Zutrittsbarrieren existieren, die einer Beteiligung am Open–Source-Entwicklungsprozess entgegenstehen, bestehen zudem Anreize, diese Verbesserungen tatsächlich durchzuführen, selbst wenn (oder weil?) es sich nur um margi-

4 Anders als die GPL enthält die BSD-Lizenz keine *Copyleft*-Klausel, d. h., dass Weiterentwicklungen auch unter eine andere Lizenz gestellt oder in proprietäre Software übernommen werden können und nicht im Quellcode veröffentlicht werden müssen.

nale Verbesserungen handelt. Aufgrund des internetbasierten Entwicklungsprozesses kann eine Vielzahl inkrementeller Entwicklungsschritte schnell zu einer signifikanten Verbesserung kumulieren.

Aufbau von Humankapital

Das Motiv, die eigenen Programmierfähigkeiten zu verbessern, wird in empirischen Untersuchungen regelmäßig als einer der wichtigsten Gründe für die Beteiligung an der Open-Source-Entwicklung benannt. Das in der Open-Source-Technologie enthaltene Wissen kann daher als öffentlich verfügbare Ressource zum Aufbau eines eigenen Humankapitals betrachtet werden, die allen offen steht und eine schnelle Weiterverbreitung ermöglicht (vgl. Pasche und von Engelhardt 2004, S. 18 f.).

Bereits das Lesen des Quellcodes, vor allem aber die Zusammenarbeit in einer weltweit vernetzten Community ermöglicht es, sich schnell und effektiv neues Wissen anzueignen und die eigenen Programmierfähigkeiten unter Anleitung und Begutachtung externer *peers* zu verbessern. Die kostenfreie Verfügbarkeit von Tools und Entwicklungswerkzeugen ermöglicht es zudem, sich mit dem Umgang dieser Instrumente vertraut zu machen und die eigene Attraktivität für einen potenziellen Arbeitgeber zu steigern.

Das in Open-Source-Software enthaltene Wissen kann als öffentliches Gut interpretiert werden, da es nicht-exklusiv und nicht-rivalisierend zugänglich ist. Durch die Entwicklung weiterer Open-Source-Software wird dieser öffentlich zugängliche Wissenskapitalstock weiter vergrößert. Die Nicht-Exklusivität dieses Wissens generiert positive externe Effekte und macht es insbesondere unabhängigen Entwicklern und kleinen Softwareunternehmen möglich, vom Wissen anderer zu profitieren. Die Möglichkeit dieser ungehinderten Weitergabe von Wissen wird durch die Quelloffenheit von Open-Source-Software sowie entsprechend ausgestaltete Nutzungslizenzen abgesichert.

Offene Schnittstellen

Standards definieren die Schnittstellen für den Austausch von Informationen und Daten zwischen verschiedenen Programmen, sie sind Voraussetzung für die Interoperabilität verschiedener Systeme auch über die Grenzen einzelner Anbieter hinweg. Aufgrund der Offenheit des Quellcodes sind bei Open-Source-Software definitionsgemäß auch alle Schnittstellen im Quellcode enthalten und dokumentiert, offen einsehbar und können diskriminierungsfrei genutzt werden.

Offene Schnittstellen verhindern, dass die technische Spezifikation eines Standards geheimgehalten und von einzelnen Anbietern exklusiv genutzt werden kann. Somit wird es auch anderen Anbietern ermöglicht und vereinfacht, komplementäre Produkte zu entwickeln, die zueinander kompatibel sind. Dies erleichtert den Marktzutritt neuer Unternehmen, welche von einem bestehenden Netzwerk auf Basis eines einheitlichen Standards profitieren können.

Bei Open-Source-Software wird durch die Nutzung offener Standards auch der Gefahr entgegengewirkt, dass es durch eine Vielzahl geschlossener Standards zu einer

Zersplitterung des Marktes kommen könnte. Eine solche Zersplitterung – man denke an die *Unix-Kriege* der achtziger Jahre, als eine nahezu unüberschaubare Anzahl verschiedenster, zueinander nicht kompatibler kommerzieller Unix-Varianten angeboten wurde – kann nur bei geschlossenen Standards auftreten, wenn die technische Spezifizierung unklar ist oder nicht genutzt werden darf und somit keine Kompatibilität hergestellt werden kann. Bei Open-Source-Software ist es hingegen möglich, Standards zu übernehmen. Somit wird der Gefahr eines technologisch bedingten Lock-in entgegengewirkt. Der beste Standard kann im Wettbewerb *entdeckt* werden und darf von anderen übernommen und in eigene Softwareprodukte implementiert werden.

3.4. Falsche Innovationsanreize im Open-Source-Modell?

Die Offenheit des Quellcodes von Open-Source-Software in Verbindung mit einer freien Lizenz ermöglicht es in besonderer Weise, innovative Produkte und Technologien zu entwickeln. Das Innovationspotenzial von Open-Source-Software knüpft aber vor allem daran an, dass bereits bestehendes Wissen uneingeschränkt genutzt werden kann, Open-Source-Software also *ex post* effizient genutzt wird. Im Gegensatz dazu bezieht sich Kritik am Open-Source-Modell vor allem auf eine Ineffizienz *ex ante*, dass also die Anreize, Open-Source-Software zu schaffen, ineffizient ausgestaltet sind. Im Hinblick auf das Innovationspotenzial von Open-Source-Software bedeutet dies, dass die Quantität („zu wenig") bzw. die Qualität („nicht marktgerecht") der hervorgebrachten Innovationen als unzureichend und dem proprietären Modell unterlegen betrachtet werden.

Der erste Kritikpunkt, dass Open-Source-Software zu wenig Innovation hervorbringen könnte, knüpft an das Fehlen exklusiver Nutzungsrechte an. Wegen der fehlenden Ausschließbarkeit von Nutzern bestehen demnach nur reduzierte Anreize, Innovationen zu schaffen, da anders als im proprietären Softwaremodell kein exklusives Recht besteht, diese Innovation vermarkten und refinanzieren zu können (Kooths et al. 2003, S. 78–80).

Dieser Überlegung steht jedoch entgegen, dass es zumindest bei Verwendung einer *Copyleft*-Lizenz (z. B. der GPL) gerade nicht möglich ist, den Programmcode von Open-Source-Entwicklungen ungehindert in eigene proprietäre Entwicklungen zu übernehmen. Eine solche Übernahme von Code erfordert es, dass darauf aufbauende Entwicklungen ebenfalls unter die Bestimmungen der *Copyleft*-Lizenz gestellt werden. Eine reine proprietäre Nutzung ist hingegen nicht möglich. Der ursprüngliche Innovator, d. h. der Entwickler der originären Open-Source-Software, kann zwar nicht verhindern, dass andere seine Entwicklung nutzen, er kann aber darauf vertrauen, umgekehrt auch von den Entwicklungen anderer zu profitieren.[5]

Der zweite Kritikpunkt, dass Innovationen im Open-Source-Modell im Vergleich zur proprietären Entwicklung und Vermarktung qualitativ schlechter seien, knüpft an die Lizenzkostenfreiheit von Open-Source-Software an. Daraus wird abgeleitet, dass Open-Source-basierte Umsätze und Erträge allein aus dem Verkauf komplementärer

5 Vgl. zum Reziprozitätsprinzip auch Ghosh (1998) und Kollock (1999).

Produkte generiert werden können. Die fehlende unmittelbare Bepreisung bedeute, dass die Open-Source-Entwicklung bewusst außerhalb von Marktmechanismen erfolge, so dass die aus dem Preismechanismus resultierende Informations-, Koordinations- und Anreizfunktion nicht genutzt wird. Dies würde bedeuten, dass den Entwicklern keine oder nur unzureichende Informationen über die Bedürfnisse der Nutzer ihrer Software vorliegen und die Allokation knapper (Entwickler-) Ressourcen sich infolgedessen nicht an den marktlichen Knappheitsrelationen orientieren kann, so dass Open-Source-Software nicht dem gesellschaftlich optimalen Verwendungszweck entspricht. Im Vergleich dazu sei die proprietäre Softwareentwicklung auf die Wünsche und Bedürfnisse der Nachfrager ausgerichtet, weil sie sich in einem funktionsfähigen Markt behaupten müssen, so dass das proprietäre Entwicklungsmodell im Ergebnis qualitativ „bessere" Innovationen hervorbringe (Kooths et al. 2003).

Diese Kritik ist zwar in sich schlüssig und weist auf eine konzeptionelle Schwäche des Open-Source-Entwicklungsmodells hin, basiert aber auf der Annahme, dass die klassische proprietäre Software in einem vollkommenen Markt entsteht. Aufgrund der zu beobachtenden Unvollkommenheiten des Softwaremarktes ist es aber fraglich, ob dem proprietären Entwicklungsmodell die vollständige Konkurrenz als Referenzmaßstab zugrunde gelegt werden kann. Vielmehr ist davon auszugehen, dass auch die Entwicklung von proprietärer Software sich nicht in vollem Umfang allein an den Kräften des Wettbewerbs orientiert (Pasche und von Engelhardt 2004).

Zum einen implizieren die spezifischen Eigenschaften von Software als digitalem Gut eine Tendenz zur Marktkonzentration: Wenige oder sogar einzelne Anbieter werden einen Markt dominieren und die überwiegende Marktnachfrage auf sich ziehen können. Die am Markt zu erzielenden Preise enthalten damit auch eine Monopolrente. Damit bestehen für einen etablierten Anbieter Anreize, sich strategisch zu verhalten, um im Rahmen der Gewinnmaximierung monopolistische Gestaltungsspielräume möglichst langfristig zu erhalten. Dies bedeutet, dass sich die Produktentwicklung nicht unbedingt am maximalen Kundennutzen, sondern möglicherweise an einer Aufrechterhaltung von Wechselkosten orientiert, um Konkurrenten den Marktzutritt zu erschweren. Beispielsweise ist zu beobachten, dass auch bei proprietärer Software auf einen Verkaufspreis verzichtet wird, wenn etwa ein Produkt im Bündel mit anderer Soft- und Hardware angeboten wird, um eine möglichst schnelle Verbreitung zu erreichen und die Kunden an ein eigenes Netzwerk zu binden. Zum anderen besteht natürlich auch im proprietären Sektor die Gefahr, dass trotz intensiver Marktforschung die Kundenbedürfnisse falsch erfasst werden, d. h., dass eine Entwicklung „am Markt vorbei" stattfindet: Nicht alles, was entwickelt wird, ist gewünscht, und nicht alles, was gewünscht ist, wird auch entwickelt.

Im Ergebnis bedeutet dies, dass auch proprietäre Entwicklungen nicht per se zu *besserer* Innovation führen. Damit wird die These, dass die im proprietären Entwicklungsmodell hervorgebrachte Innovation von höherer Qualität sei als im Open–Source-Modell, zumindest abgeschwächt.

4. Zusammenfassung

Im vorliegenden Beitrag wurde zunächst ein theoretischer Bezugsrahmen für die Analyse des Wettbewerbs auf Softwaremärkten entwickelt. Die ökonomischen Eigenschaften von Software als digitalem Gut implizieren die Existenz von angebots- und nachfrageseitigen Größenvorteilen, welche die empirisch zu beobachtende Tendenz zur Marktkonzentration in Softwaremärkten erklären können.

Dem steht als Korrektiv eine hohe Dynamik des technischen Wandels auf den Märkten der Informations- und Kommunikationstechnologie entgegen. Dessen Bedeutung wird unterstrichen durch die Aussagen der Theorie bestreitbarer Märkte, wonach Marktanteile nur bedingt aussagekräftig zur Beurteilung der Wettbewerbsintensität sind. Open-Source-Software trägt dazu bei, den tatsächlichen sowie den potenziellen Wettbewerb auf Softwaremärkten zu erhöhen. Voraussetzung ist jedoch, dass keine bedeutsamen Marktbarrieren bestehen, die den Zutritt neuer Unternehmen erschweren oder gar verhindern.

Da Open-Source-Software die spezifischen Eigenschaften digitaler Güter anders nutzt, werden zusätzlich andere Präferenzen der Marktteilnehmer abgedeckt. Aufgrund der spezifischen Eigenschaften von Open-Source-Software, d. h. der Offenheit des Quellcodes in Verbindung mit einer freiheitlichen Nutzungslizenz, ergeben sich auch spezifische Innovationspotenziale.

Als öffentlich verfügbare Ressource mit niedrigen Markteintrittsbarrieren dient Open-Source-Software als technische Basis für neue Produkte und kann als kostengünstiger Inputfaktor betrachtet werden, von dem Dritte nicht ausgeschlossen werden können. Weiterentwicklungen sind oftmals nutzerinduziert, inkrementelle Entwicklungsschritte werden vereinfacht. Open-Source-Technologie kann auch als öffentliche Ressource zum Aufbau von Humankapital betrachtet werden, welche als öffentliches Gut nicht-exklusiv und nicht-rivalisierend zugänglich ist und damit den öffentlich zugänglichen Wissenskapitalstock vergrößert. Aufgrund der Offenheit von Schnittstellen können Innovations- und Wettbewerbspotenziale über die Grenzen einzelner Anbieter und Systeme hinweg genutzt werden. Dies erleichtert den Marktzutritt neuer Unternehmen und wirkt auch der Gefahr einer Zersplitterung aufgrund inkompatibler geschlossener Standards entgegen.

Kritik am Innovationspotenzial von Open-Source-Software knüpft an das Fehlen exklusiver Nutzungsrechte („zu wenig Innovation“) und den mit der Lizenzkostenfreiheit verbundenen Verzicht auf die Preisfunktionen des Marktes („zu schlechte Innovation“) an. Allerdings ist zu fragen, ob das Modell der vollständigen Konkurrenz den geeigneten Vergleichsmaßstab darstellt oder ob nicht auch die proprietäre Softwareentwicklung durch ähnliche Unvollkommenheiten gekennzeichnet ist.

Literaturverzeichnis

Baumol, W. J. (1982), 'Contestable Markets: An Uprising in the Theory of Industry Structure', *American Economic Review* **71**(1), S. 1–15.

Baumol, W. J., Panzar, J. C. und Willig, R. D. (1988), *Contestable Markets and the Theory of Industry Structure*, Überarbeitete Aufl., Harcourt Brace Jovanovich, San Diego.

Ben-Shahar, D. und Jacob, S. (2001), Preach for Breach: Selective Enforcement of Copyrights as an Optimal Monopolistic Behavior. Vortrag beim „Workshop on the Microeconomics of the New Economy" Institut für Weltwirtschaft, Kiel, September 2001.

Ghosh, R. A. (1998), 'Cooking Pot Markets: An Economic Model for the Trade in Free Goods and Services on the Internet', *First Monday* **3**(3). http://www.firstmonday.org/issues/issue3_3/ghosh/.

Ghosh, R. A., Glott, R., Krieger, B. und Robles, G. (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.

Gröhn, A. (1999), *Netzwerkeffekte und Wettbewerbspolitik: Eine ökonomische Analyse des Softwaremarktes*, Vol. 296 of *Kieler Studien*, Mohr Siebeck, Tübingen.

Jaeger, T. und Metzger, A. (2002), *Open Source Software: Rechtliche Rahmenbedingungen der Freien Software*, C.H. Beck, München.

Katz, M. L. und Shapiro, C. (1999), Antitrust in Software Markets, *in* J. A. Eisenach und T. M. Lenard (Hrsg.), 'Competition, Innovation, and the Microsoft Monopoly: Antitrust in the Digital Market', Kluwer Academic Publishers, Boston.

Klodt, H., Buch, C. M., Christensen, B., Gundlach, E., Heinrich, R. P., Kleinert, J., Piazolo, D., Sailer, K. und Stehn, J. (2003), *Die neue Ökonomie: Erscheinungsformen, Ursachen und Auswirkungen*, Vol. 321 of *Kieler Studien*, Springer, Berlin.

Kollock, P. (1999), The economies of online cooperation: Gifts and public goods in cyberspace, *in* M. A. Smith und P. Kollock (Hrsg.), 'Communities in Cyberspace', Routledge, London, Kapitel 9, S. 220–39.

Kooths, S., Langenfurth, M. und Kalwey, N. (2003), 'Open Source-Software: Eine volkswirtschaftliche Beurteilung', *MICE Economic Research Studies* **4**. Muenster Institute for Computational Economics, Westfälische Wilhelms-Universität Münster. http://mice.uni-muenster.de/mers/mers4-OpenSource_de.pdf.

Liebowitz, S. J. und Margolis, S. E. (1999), *Winners, losers & Microsoft*, The Independent Institute, Oakland, CA.

Mendys-Kamphorst, E. (2002), 'Open vs. Closed: Some Consequences of the Open Source Movement for Software Markets', CPB Discussion Paper No. 13. CPB Netherlands Bureau for Economic Policy Analysis, The Hague.

Monopolkommission (2003), 'Netzwettbewerb durch Regulierung', Drucksache des Deutschen Bundestages und -rates sowie Nomos, Baden-Baden. Vierzehntes Hauptgutachten, gemäß Par. 44 Abs. 1 Satz 1 GWB, http://www.monopolkommission.de/haupt.htm.

Pasche, M. und von Engelhardt, S. (2004), 'Volkswirtschaftliche Aspekte der Open-Source-Softwareentwicklung', Jenaer Schriften zur Wirtschaftswissenschaft 18/2004. Wirtschaftswissenschaftliche Fakultät, Friedrich-Schiller-Universität Jena. http://www.wiwi.uni-jena.de/Papers/wp-sw1804.pdf.

Quah, D. (2003), Digital Goods and the New Economy, *in* D. C. Jones (Hrsg.), 'New Economy Handbook', Elsevier Academic Press, San Diego, CA.

Raymond, E. S. (1998), 'The cathedral and the bazaar', *First Monday* **3**(3). http://www.firstmonday.org/issues/issue3_3/ghosh/.

Rosenberg, D. K. (2000), *Open Source: The Unauthorized White Papers*, IDG Books Worldwide, Foster City, CA.

Sachverständigenrat zur Begutachtung der Gesamtwirtschaftlichen Entwicklung (2000), 'Chancen auf einen höheren Wachstumspfad', http://www.sachverstaendigenrat-wirtschaft.de/gutacht/verzeich.html und bei Verlag Metzler-Poeschel, Reutlingen. Jahresgutachten 2000/2001.

Sailer, K. (2001), 'Regulierungsbedarf in Netzwerken? Implikationen für die Internetökonomie', *Die Weltwirtschaft* **4**, S. 350–378.

Shapiro, C. und Varian, H. R. (1999), *Information Rules: A Strategic Guide to the Network Economy*, Harvard Business School Press, Boston.

Varian, H. R. (2000), Markets for Information Goods, *in* 'Monetary Policy in a World of Knowlege-Based Growth, Quality Change, and Uncertain Measurement'. Prepared for Bank of Japan conference, June 18-19, 1998. http://www.sims.berkeley.edu/~hal/Papers/japan/japan.pdf.