Open Source Jahrbuch 2008

Zwischen freier Software und Gesellschaftsmodell

Herausgegeben von Bernd Lutterbeck, Matthias Bärwolff und Robert A. Gehring

Open Source Jahrbuch 2008

Zwischen freier Software und Gesellschaftsmodell

Herausgegeben und bearbeitet von:

Bernd Lutterbeck
Professor für Informatik und Gesellschaft an der TU Berlin, Professor der Aktion Jean Monnet für europäische Integration

Matthias Bärwolff
Wissenschaftlicher Mitarbeiter und Doktorand am Fachgebiet Informatik und Gesellschaft an der TU Berlin

Robert A. Gehring
Doktorand am Fachgebiet Informatik und Gesellschaft an der TU Berlin

Daniel Auener
Student der Informatik, TU Berlin

Richard Bretzger
Student der Soziologie technikwissenschaftlicher Richtung, TU Berlin

Matthias Choules
Student der Informatik, TU Berlin

Bob Dannehl
Student der Informatik, TU Berlin

Mathias Eberle
Student der Informatik, TU Berlin

Roman Rauch
Student der Informatik, TU Berlin

Marc Schachtel
Student der Informatik, TU Berlin

Matthias Schenk
Student der Informatik, TU Berlin

Malte Schmidt-Tychsen
Student der Betriebswirtschaft, TU Berlin

2008

Lehmanns Media LOB.de

Bibliografische Informationen der Deutschen Bibliothek:
Die Deutsche Bibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet unter **<http://dnb.ddb.de>** abrufbar.

Open Source Jahrbuch 2008
Zwischen freier Software und Gesellschaftsmodell
Lutterbeck, Bernd; Bärwolff, Matthias; Gehring, Robert A. (Hrsg.)

Berlin, 2008 – Lehmanns Media – LOB.de
ISBN: 978–3–86541–271–3

Satz und Layout: Mathias Eberle, Matthias Liebig und Wolfram Riedel
unter Verwendung des Textsatzsystems LaTeX

Einbandgestaltung: Matthias Bärwolff, Nadim Sarrouh und Marc Schachtel
unter Verwendung des Textes der *GNU General Public License*

Druck und Verarbeitung: Druck- und Verlagsgesellschaft Rudolf Otto mbH, Berlin

Dieses Buch wurde auf chlorfrei gebleichtem Recycling-Papier gedruckt.

Inhalt

Vorwort der Herausgeber IX
von Bernd Lutterbeck, Matthias Bärwolff und Robert A. Gehring

Kapitel 1 Vom Anwender zum Entwickler

Das Zeitalter der altruistischen Partizipation 3
von Bob Dannehl

Anwenderemanzipation – Wie Nutzer die Softwareentwicklung beeinflussen können. 7
von Jan Ulrich Hasecke

Architecture of Participation: Teilnehmende Open Source bei MySQL 19
von Kaj Arnö

OpenOffice.org – Aus dem Alltag eines nicht alltäglichen Open-Source-Projekts 27
von Jacqueline Rahemipour

OpenMoko – Freie Software für Mobiltelefone 41
von Michael Lauer

Endanwendergetriebene Open-Source-Softwareentwicklung mit Cofundos . . 51
von Sören Auer

Kapitel 2 Von der Innovation zum Geschäftsmodell

Open Source als Werkzeug der Wirtschaft 63
von Matthias Choules und Roman Rauch

Monopolelemente bei freier Software 65
von Matthias Bärwolff

Unternehmen zwischen Offenheit und Profitstreben 77
von Joel West

OS-Marketing – Warum Konsumenten am Marketing von Unternehmen teilnehmen 91
von Klaus-Peter Wiedmann, Lars Pankalla und Sascha Langner

Der Fall Microsoft – Offengelegte Schnittstellen und offengebliebene Fragen . 105
von Leonie Bock

Kapitel 3 Von der digitalen Herausforderung zum sozialen Prozess

Open Source trägt Verantwortung 121
von Richard Bretzger

Vom Revolutionär zum Unternehmer – Die F/OSS-Bewegung im Wandel . . 123
von Andrea Hemetsberger

Digitale Produktionsgemeinschaften: Open-Source-Bewegung als deterritoriale Vergemeinschaftung 135
von Daniel Tepe und Andreas Hepp

Kapitel 4 Von der Entscheidung zum Einsatz

Auf der Suche nach Mitteln, die den Zweck heiligen 153
von Marc Schachtel und Malte Schmidt-Tychsen

Die Bedeutung von Open Source in der öffentlichen Verwaltung und der IT-Branche 155
von Jochen Günther

Messung von Offenheit an IT-Artefakten 169
von Jan Suhr

Die deutsche Rechtsprechung zur GNU General Public License 185
von Henriette Picot

Open Source Content Management – Eine kritische Betrachtung 197
von Tobias Hauser und Andi Pietsch

Change Management: Linux-Desktop-Migration mit Erfolg. 207
von Beate Groschupf und Natascha Zorn

Kapitel 5 Vom Wissen zur Vernetzung

Fest im Griff – Netze und Menschen 219
von Mathias Eberle

Richard Stallmans Goldene Regel und das „Digital Commons“ 223
von Glyn Moody

Archivierung und Open Source 233
von Christoph Jeggle und Ulrich Kampffmeyer

Linux4Afrika – Ein Linux-Terminal-Server-Projekt. 249
von Hans-Peter Merkel

Glossar. 259

Stichwortverzeichnis 267

Mitwirkende 273

Das Open-Source-Jahrbuch-Projekt 279

Lizenzen 281

Vorwort der Herausgeber

Bernd Lutterbeck, Matthias Bärwolff
und Robert A. Gehring

Nach fünf Jahren Open-Source-Jahrbuch kommt man kaum umhin, einen Rückblick zu wagen auf die Zeit, die vergangen ist, und die Erfahrungen, die wir gesammelt haben. Haben wir unsere Ziele erreicht? Diese Frage ist in der Tat gar nicht so einfach zu beantworten, schließlich haben wir den Luxus, ein akademisches Projekt unter dem Dach eines gemeinnützigen Vereins zu sein, das niemandem Rechenschaft schuldig ist. Wenn das primäre Ziel also nicht die Mehrung von finanziellen Werten war, was war dann überhaupt das Ziel? Vielleicht sollten wir uns dieser Frage nähern mit einem Trick aus dem akademischen Leben: Ziel und Methode einer Arbeit werden nicht vor, sondern nach Fertigstellung sozusagen abgeleitet aus dem, was geworden ist.

Also, was ist geworden? Die Trivialitäten zuerst: fünf Open-Source-Jahrbücher, 158 Artikel, circa 1500 verkaufte Exemplare, hunderttausende Downloads von unserer Webseite. Was die abrechenbaren Einnahmen und Ausgaben angeht: circa 30 000 Euro Einnahmen durch Buchverkäufe und 5700 Euro durch Spenden; dazu Ausgaben in Höhe von circa 27 000 Euro für den Druck der Bücher, circa 3000 Euro für das Lektorat ab dem 2005er Jahrbuch und 5000 Euro sonstige Spesen, überwiegend durch unsere CeBIT-Beteiligungen. In fünf Jahrbuch-Projektgruppen haben insgesamt circa 25 Studenten in der Redaktion gearbeitet, mindestens ein Herausgeber war zumindest teilweise in die redaktionelle Arbeit eingebunden. Der gesamte Arbeitsaufwand der Projektgruppen entspricht damit in etwa einer vollen Stelle – Bruttokosten also von circa 250 000 Euro, die uns die Studenten erspart haben. Dafür an dieser Stelle unser Dank an die Studenten!

Haben wir aber, wie der Engländer so schön sagt, „einen Unterschied gemacht", unsere Mühen also irgendeine positive Auswirkung gehabt? Hier gibt es sicher zwei Ebenen zu betrachten, zum einen die dem eigentlichen Projekt endogene und zum anderen die dem Projekt exogene. Auf ersterer sind zweifellos wichtige Erfahrungen gemacht und Kompetenzen gesammelt worden, mal mehr, mal weniger. Das Muster folgt ziemlich genau der Gauß'schen Normalverteilung, wenige Studenten haben das

Projekt geprägt und vorangebracht im besten Sinne des Wortes, viele haben gut mitgearbeitet und einige wenige wiederum haben mehr Arbeit verursacht als erledigt.

Was den exogenen Aspekt angeht, ist es schwierig, klare Aussagen zu treffen. Bei *Google Scholar* findet man circa ein Dutzend wissenschaftliche Artikel, die auf das Jahrbuch verweisen. Das ist nicht viel. Andererseits muss die wissenschaftliche Debatte, die sich auf das Jahrbuch stützt, nicht unbedingt Gradmesser sein für unseren Erfolg. Wissenschaft ist ja nicht selten auch selbstreferenziell und letztlich irrelevant, auf der Suche nach der Wahrheit werden viele Pfade verfolgt, auch solche, die ins Leere führen.

Die allermeisten unserer Leser sind jedoch gar keine Wissenschaftler, sondern mittelständische Unternehmer oder einfach nur interessierte Privatleute – das wissen wir durch die Bestellungen über unsere Webseite opensourcejahrbuch.de. Wenn es am Anfang des Projekts ein erklärtes Ziel gab, dann also auch dieses: das gesamte Spektrum der Debatte um Open Source und verwandte Themen abdecken, inklusive praktischer und esoterischer Aspekte. So erklärt sich wohl auch, warum neben praxisnahen Beiträgen gerade die unwissenschaftlichsten Artikel des Buchs, die polemischsten mithin, die erfolgreichsten waren. Etwa solche zu freier Software und freier Gesellschaft oder freier Software und der digitalen Kluft zwischen Nord und Süd.

Nicht um Wissenschaft als solche geht es zuvörderst im Buch, sondern darum, diese einer breiten Masse zu vermitteln. Die Downloads der Bücher und der einzelnen Artikel bewegen sich im sechsstelligen, die Verkäufe aller Ausgaben zusammen nur im unteren vierstelligen Bereich. Auch das muss also ein Ziel gewesen sein: Wissen ohne jede Diskriminierung für jeden interessierten Leser verfügbar zu machen und dies möglichst zu jedem Zwecke, ob nichtkommerziell oder kommerziell. Die allermeisten Autoren konnten wir in diesem Sinne für eine Lizenzierung ihrer Texte unter Creative-Commons-Lizenzen gewinnen, die eine solch breite Nutzung ermöglichen.

Damit hätten wir die Ziele identifiziert, zumindest die, die wir erreicht zu haben glauben: das Thema Open Source erschöpfend und allgemeinverständlich zu bearbeiten und an einen breiten Leserkreis zu vermitteln. Schließlich noch ein Wort zur Methode. Im Prinzip haben wir uns die Sache einfach gemacht, gerade beim vorliegenden Buch. Wir als Herausgeber haben diesmal spürbar weniger in den Redaktionsbetrieb, die Auswahl der Autoren und die Bearbeitung der Texte eingegriffen. Dies haben unsere Studenten unter der Leitung von Daniel Auener erledigt. Es bleibt uns nur zu sagen: Hut ab!

Und natürlich, viel Spaß beim Lesen.

Prof. Dr. iur. Bernd Lutterbeck
Matthias Bärwolff
Robert A. Gehring

Kapitel 1

Vom Anwender zum Entwickler

„Wer heute nur immer das tut, was er gestern schon getan hat, der bleibt auch morgen, was er heute schon ist."

– Nils Goltermann

Das Zeitalter der altruistischen Partizipation

Bob Dannehl

> „Reale soziale Bindungen gehen zu Bruch, weil der [Internet-]Nutzer sich immer mehr in die virtuelle Welt hineinversetzt, in der zwischenmenschliche Kontakte vermeintlich nur per Knopfdruck bequem auf- und wieder abgebaut werden können." (Besim Karadeniz)[1]

Das Internet-Zeitalter wird häufig mit einem Verlust sozialer Kompetenzen und der geistigen Vereinsamung des Einzelnen in Verbindung gebracht. In einigen Fällen mag das sicher zutreffen, aber die Mehrheit scheint den virtuellen Raum als *das* Medium angeregten Wissensaustauschs überhaupt zu begreifen und entsprechend zu nutzen. Menschen treten im mondialen Netzwerk, über alle Grenzen hinweg, mit anderen in Kontakt.

Im Meer der Möglichkeiten, die uns das Internet bietet, ist eine ganz besonders wichtig i Bezug auf Open Source: verzugsfreie, globale Kommunikation. Erst sie ebnete den Weg für Partizipation im Bereich von Open-Source-Software, bekanntestes Beispiel: Linus Torvalds. Ohne das Internet würde er womöglich noch heute allein an seinem PC in Helsinki sitzen, als einziger Linux-Nutzer der Welt. . .

Die Idee über das Internet Mitstreiter und gemeinsame Lösungen zu finden, hat ganz neue Formen der Kooperation entstehen lassen. Heute ist es problemlos möglich, weltweit am selben Projekt zu wirken, auch wenn dies bedeuten kann, Weggefährten eventuell niemals zu Gesicht zu bekommen.

Diese neue Art der Kommunikation über das Internet vereinfacht aber nicht nur die kooperative Entwicklung von Open-Source-Software, indem es die Entwickler vernetzt, sondern ermöglicht es auch den Anwendern, ihre Wünsche zu kommunizieren und gegebenenfalls selbst zu Entwicklern zu werden.

Solche global agierenden Funktionsgemeinschaften, die alle an einer bestimmten Software interessierten Gruppen vereinigen, lassen sich im Open-Source-Umfeld immer häufiger beobachten und erhalten kontinuierlichen Zulauf.

1 Siehe http://www.netplanet.org/netlife/life001.shtml [09. Feb. 2008].

Die Effizienz und Antriebskraft dieser Projekte erscheinen für Außenstehende oftmals verblüffend. Wie findet die Koordination, wie Planung oder auch Einflussnahme der Anwender in diesen Projekten statt? Warum engagieren sich so viele, um ohne monetäre Entlohnung an der Entwicklung von Software mitzuwirken?

Heutzutage, wo Effizienzsteigerung häufig an erster Stelle zu stehen scheint und der Mensch oftmals lernen muss, sich auf das Wesentliche zu beschränken, stellen Open-Source-Projekte mit ihrer Vielzahl freiwilliger Helfer ein absolutes Phänomen dar. Geradezu paradox erscheint es, wie Menschen im schnelllebigen und hastigen IT-Zeitalter immensen Zeitaufwand betreiben, um in den zahlreichen Open-Source–Communitys aktiv zu werden und mit ihren Beiträgen das große Ganze sukzessive voranzubringen. Faszinierend ist vor allem die Tatsache, dass die Beiträge letztendlich aus unterschiedlichsten Motivationen sowie eher *persönlichen Interessen* heraus entstehen und dennoch die Bausteine eines selbstlos erscheinenden *Gemeinschaftswerks* bilden.

Obwohl der Einfluss namhafter Firmen innerhalb zahlreicher großer Projekte merklich stärker geworden ist, kann man dennoch nur bedingt von zunehmender Kommerzialisierung sprechen. Die freiwillige Partizipation an Open-Source-Projekten ist noch immer von existenzieller Bedeutung und sehr häufig ein entscheidender Faktor des Unternehmenserfolgs. Dabei ist es offenkundig nicht zwingend erforderlich, über Programmierkenntnisse zu verfügen. Es gibt viele Wege, Einfluss auf das Geschehen zu nehmen, beispielsweise durch das Erstellen von Fehlerberichten, Kommentierung des Programmierfortschritts oder indem man anderen Hilfe auf den diversen Mailinglisten anbietet.

Das folgende Kapitel gibt einen breiten Überblick zu Wegen, Zielen und Motivationen von Open-Source-Projekten, wie beispielsweise *MySQL* oder auch *OpenOffice.org*. Besonderes Augenmerk liegt dabei auf der Einbindung von Anwendern in den Entwicklungsprozess.

Den Anfang macht Jan Ulrich Hasecke. In seinem Artikel skizziert er am Beispiel des Webapplikationsservers *Zope* die Möglichkeiten des Einzelnen, auf Open–Source-Produkte und die dahinterstehende Entwicklung Einfluss zu nehmen, um somit Software an die eigenen Bedürfnisse anzupassen. Seinen Fokus legt er dabei auf die nötigen Voraussetzungen: zum einen die Veranlagung des Anwenders, zum anderen bestimmte Eigenschaften, die die Software selbst beziehungsweise ihr Umfeld im Allgemeinen mit sich bringen sollte, um die aktive Teilnahme der Anwender zu begünstigen.

Kaj Arnö prägt in seinem Artikel den Begriff der „Teilnehmenden Entwicklung" am Beispiel von *MySQL*, der wohl bekanntesten Open-Source-Datenbank unserer Zeit, die zuletzt im Januar 2008 im Zuge der Übernahme durch *Sun Microsystems* Aufsehen erregte. Detailliert beschreibt er, wie sich die Zusammenarbeit mit interessierten Entwicklern außerhalb der Firma von einer „stiefmütterlichen" Handhabung hin zu einem immer wichtigeren Teil der Gesamtentwicklung gemausert hat. Interessante Analogien zu traditionellen skandinavischen und finnischen Kooperationsmodellen machen

das Geschriebene zudem außerordentlich anschaulich und zeigen, dass es mitunter ein langwieriger Prozess sein kann, das gesamte Potenzial der eigenen Community sinnvoll auszuschöpfen.

Im Anschluss gewährt uns Jacqueline Rahemipour von *OpenOffice.org* interessante Einblicke in die Welt dieses faszinierenden Projekts. Im Zuge vielfältiger Einflüsse und einer starken Dynamik aus der Community ist es *OpenOffice.org* gelungen, eine wirkliche Alternative zu proprietären Office-Suiten zu etablieren. Frau Rahemipour geht dabei zunächst auf die Entstehungsgeschichte ein, bevor sie Aspekte wie Organisationsstrukturen, Community-Building und Marketing vorstellt. Aber auch brisante Themen wie z. B. der Frauenanteil bei *OpenOffice.org* und in Open-Source-Projekten im Allgemeinen werden aufgegriffen.

Dass es auch für Mobiltelefone Alternativen zu der proprietären Betriebssoftware der Hersteller gibt, zeigt Michael Lauer in seinem Beitrag über *OpenMoko*. Dabei handelt es sich um eine freie Software-Plattform für Mobilkommunikation, die es ermöglicht, weitaus mehr als nur die üblichen, stark von der Hardware gekapselten JAVA-Anwendungen auszuführen. Da es ein Community-basierter Standardisierungsansatz ist und interessierte Entwickler somit eigene Korrekturen oder neue Funktionalitäten einpflegen können, darf sich jedermann herzlich eingeladen fühlen, an der „Befreiung der Mobiltelefone" mitzuwirken.

Aus einer etwas anderen Richtung als die bisherigen Beiträge beleuchtet Sören Auer die Möglichkeiten jedes Einzelnen, Software den eigenen Erfordernissen anzupassen. Möchte man ein dringend benötigtes Feature einer Software zeitnah realisieren, verfügt aber nicht selbst über das nötige Know-how beziehungsweise entsprechende Geldressourcen, um Dritte zu beauftragen, bietet die Open-Source-Innovationsplattform *Cofundus* eine mögliche Alternative der Umsetzung. Vereinfacht ausgedrückt, funktioniert dieses Konzept als eine Art Ausschreibung für Open-Source-Software, wobei Personen, die an der gleichen zu entwickelnden Funktionalität interessiert sind, gemeinsam dafür Geld spenden, bis ein jeweiliger Schwellwert erreicht ist und ein geeigneter Entwickler mit der Umsetzung beauftragt werden kann. Mit Auers Artikel schließt dieser Abschnitt des Buchs im Wissen, dass auch Anwendern, die sich nicht einbringen können oder möchten, keinesfalls die Hände im Umgang mit freier Software gebunden sind.

Vielleicht kann dieses Kapitel dazu beitragen, die Beweggründe der vielen Open–Source-Mitstreiter zu verstehen, vielleicht sogar dazu führen, sich selbst zu engagieren. Zumindest zeigt es aber, dass freie Software wandlungsfähig ist und es viele Wege gibt, sie den eigenen Bedürfnissen anzupassen. Die dahinterstehenden Projekte und Initiaven sind bereit, das bisher Erreichte mit der Welt zu teilen und offen für neue Einflüsse. Jeder ist willkommen, um mit innovativen Ideen die Gemeinschaft zu bereichern und seinen Platz zu finden im „Zeitalter der Partizipation"...

Anwenderemanzipation – Wie Nutzer die Softwareentwicklung beeinflussen können

JAN ULRICH HASECKE

Obwohl Open-Source-Communitys gemeinhin als *user groups*, also als Gemeinschaften von Softwarenutzern gesehen werden, ist die Einbindung der Anwender in den Entwicklungsprozess von Open-Source-Software nicht *per se* gewährleistet. Die meisten Anwender sind weit davon entfernt, direkten Einfluss auf die Entwicklung der Software zu nehmen, obwohl Open–Source-Software zahlreiche Mitwirkungsmöglichkeiten bietet. Am nächsten kommen dem Idealbild des emanzipierten Anwenders, der die Entwicklung von Software maßgeblich beeinflusst, innovative Anwendernetzwerke wie *PloneGov*. In diesem Netzwerk haben sich öffentliche Verwaltungen zusammengeschlossen, um Funktionen in das Content Management System *Plone* zu integrieren, die sie in ihrem Alltag benötigen.

Schlüsselwörter: Anwenderemanzipation · Anwendernetzwerke · PloneGov · Zope · DZUG e.V. · Partizipation · Entwickler-Community

1 Einleitung

Wenn man Anwender fragt, warum sie Open-Source-Software einsetzen, werden ganz unterschiedliche Begründungen genannt. Während für den Privatanwender zumeist die lizenzkostenfreien Nutzungsrechte den Ausschlag geben, sind kommerziellen Anwendern andere Vorteile wichtiger. Je intensiver man eine Software nutzt, umso entscheidender wird der unbeschränkte Zugang zum Quellcode. Denn nur so hat der Anwender die Möglichkeit, auf die Funktionen der Software Einfluss zu nehmen. Da Open-Source-Code von jedermann verändert werden darf, kann der Anwender die Software seinen individuellen Bedürfnissen anpassen. Er kann sie umschreiben und mit anderen Softwarekomponenten kombinieren. Daher ist im Zusammenhang mit Open-Source-Software häufig die Rede vom emanzipierten Anwender.

Emanzipation ist ein weiter Begriff. Wir möchten ihn daher für den Gang unserer Überlegungen einschränken und beschreiben, welche Möglichkeiten Anwender haben, auf die Entwicklung freier Software Einfluss zu nehmen, und aufzeigen, welches emanzipatorische Potenzial in diesen Formen der Anwenderbeteiligung liegt. Anhand konkreter Beispiele soll dabei geklärt werden, wie Open-Source-Communitys den Anwender in den Entwicklungsprozess einbeziehen und inwiefern die Software selbst zur Emanzipation des Anwenders beiträgt. Es sollen einerseits der Typus des emanzipierten Anwenders skizziert und andererseits einige Eckpunkte herausgearbeitet werden, die das emanzipatorische Potenzial von freier Software ausmachen und fördern.

2 Mitwirkungsmöglichkeiten in der Community

Beginnen wir mit den traditionellen Möglichkeiten, die Open-Source-Anwendern zur Verfügung stehen, um die Weiterentwicklung von Open-Source-Software zu beeinflussen. Eng verflochten mit der Entstehung des Internets sind Mailinglisten und Usenetgruppen, in denen sich – anfangs vor allem technikaffine – Softwarenutzer und -entwickler ausgetauscht haben. Heutzutage trifft man in zahllosen Mailinglisten, Newsgruppen und Internetforen Anwender mit den unterschiedlichsten technischen Fähigkeiten. Sie nutzen die Plattformen, um ihre Wünsche nach neuen Funktionen zu äußern und mit Entwicklern zu diskutieren. Die öffentlichen Foren dienen nicht bloß dem besseren Verständnis der Software, sondern geben den Entwicklern auch wichtige Anstöße für Verbesserungen und Erweiterungen der jeweiligen Software. Das gilt nicht nur für freie Software, sondern auch für proprietäre, kommerzielle Software und die so genannten Shareware-Programme.

Bei proprietärer Software und Shareware-Programmen gibt es jedoch eine strikte Rollenverteilung zwischen Entwicklern, die allein Zugang zum Quellcode besitzen, und Anwendern, denen dieser Zugang verwehrt ist. Bei freier Software ist das anders. Nutzer mit Programmierkenntnissen können jederzeit durch eigenen Code zur Lösung eines Problems beitragen. Dadurch kann in populäre Open-Source-Software innerhalb kurzer Zeit sehr viel Know-how einfließen. Offene Software ist akkumuliertes Wissen, das der gesamten Gesellschaft zur Verfügung steht. Open-Source-Communitys werden deshalb auch als Wissensgemeinschaften beschrieben (Barcellini, Détienne und Burkhardt 2006).

Bug reports und *feature requests* sind weitere Möglichkeiten, mit denen Anwender die Softwareentwicklung beeinflussen können. In vielen Softwarepaketen, ob proprietär oder frei, gibt es bequeme Möglichkeiten, den Hersteller per E-Mail über Fehler zu informieren (*bug report*) oder den Wunsch nach neuen Funktionen (*feature requests*) zu äußern. Mit jedem Programmabsturz erhalten heutzutage die großen proprietären Softwarehersteller über die in den Betriebssystemen eingebauten Werkzeuge ein Absturzprotokoll, das es ihnen ermöglicht, den Fehler in der Software zu identifizieren und zu beheben. Doch während bei proprietärer Software die Benachrichtigungen

ausschließlich beim Hersteller eintreffen, werden sie bei freier Software in der Regel veröffentlicht. So gibt es für die meisten Open-Source-Projekte öffentliche Fehlerdatenbanken, die über das Internet zugänglich sind und in denen jeder mitverfolgen kann, welche Verbesserungsvorschläge es gibt, welche Fehler gemeldet und wie sie behoben worden sind. Fehlt eine solche öffentliche Fehlerdatenbank, wie das bei proprietärer Software in der Regel der Fall ist, weiß der Anwender weder, dass ein bestimmter Fehler entdeckt worden ist, noch kann er nachvollziehen, welche Schritte unternommen worden sind, um den Fehler zu beheben. Bei freier Software dagegen kann der Anwender nicht nur selbst einen Vorschlag zur Behebung von Fehlern einbringen, er kann auch die Diskussion über die Qualität der Fehlerbehebung auf öffentlichen Plattformen mitverfolgen und sein Handeln danach ausrichten.

Der Fahrplan, nach dem sich die Programmierer bei der Weiterentwicklung von Open-Source-Software richten, wird schon lange nicht mehr dem Zufall oder den Vorlieben einzelner Entwickler überlassen. Insbesondere die größeren Communitys haben formalisierte Steuerungsmethoden ausgebildet (siehe z. B. Brand und Schmid 2006, S. 127 ff.). Die Weiterentwicklung der objektorientierten Programmiersprache Python wird beispielsweise durch so genannte *Python Enhancement Proposals* (*PEP*) strukturiert. Ein *PEP* enthält neben Sinn und Zweck des Vorschlags eine genaue Spezifikation und einen Verweis auf eine beispielhafte Implementierung. Für das auf Zope und Python basierende Content Management System (CMS) Plone hat sich der Begriff *PLIP* (*Plone Improvement Proposal*) eingebürgert. *PEPs* und *PLIPs* durchlaufen einen strukturierten Prozess der internen Prüfung, bis sie offiziell angenommen und umgesetzt werden.

Durch öffentliche Mailinglisten, *bugtracker*, *code repositories* und *PEPs* beziehungsweise *PLIPs* entsteht ein transparenter Entwicklungsprozess, der als Interaktion in den drei Bereichen *Implementierungsbereich* (Code-Archiv und Versionierungsmechanismus), *Dokumentationsbereich* (Websites) und *Diskussionsbereich* (Mailinglisten, Newsgruppen, Weblogs, Chats) beschrieben wird.[1] Der Open-Source-Anwender, in unserem Beispiel der Nutzer der Programmiersprache Python oder des CMS Plone, verfügt damit über wesentlich mehr Informationen als der Anwender proprietärer Software, bei der sämtliche Entscheidungen über die Zukunft der Software hinter verschlossenen Türen in den Firmenzentralen der Hersteller fallen. Wie komplex das Zusammenspiel der Akteure im Entwicklungsprozess von Open-Source-Software sein kann, wurde anhand des *PEP* 279 in Barcellini et al. (2005) und des *PEP* 327 in Barcellini, Détienne und Burkhardt (2006) analysiert.

Die Existenz dieser offenen Interaktionsbereiche allein reicht natürlich nicht aus, um einen transparenten und im Idealfall vom Anwender getriebenen Entwicklungsprozess zu gewährleisten. Wird eine Open-Source-Community beispielsweise von einem oder mehreren großen Unternehmen dominiert, können unternehmensinter-

1 Siehe Barcellini, Détienne und Burkhardt (2006), Barcellini et al. (2005), Sack et al. (2006) und Barcellini, Détienne, Burkhardt und Sack (2006).

ne, informelle und intransparente Entscheidungsprozesse neben den öffentlichen entstehen.[2]

3 Chancen und Risiken von Eigenentwicklungen

Jedem Anwender steht es frei, selbst aktiv zu werden und Open-Source-Software seinen individuellen Bedürfnissen anzupassen, indem er die Software umschreibt oder bei fehlenden Programmierkenntnissen beziehungsweise mangelnden Kapazitäten einen Dienstleister damit beauftragt. Dies hat Vor- und Nachteile. Er kann zwar, je nachdem, um welche Art von Anwendung oder Software es sich handelt, die Funktionen *seiner* Individualsoftware selbst bestimmen. Er riskiert aber, seine eigene Entwicklung abseits des Hauptentwicklungsstrangs in einem so genannten Fork[3] zu betreiben, sodass die Gefahr besteht, dass er von der allgemeinen Entwicklung der Software abgekoppelt wird. Er tut daher gut daran, seine individuelle Lösung in generischer Form wieder in die Community zurückfließen zu lassen, um die Chance zu wahren, dass seine Veränderungen in den Hauptentwicklungsstrang aufgenommen werden oder sich andere Benutzer finden, die ähnliche Aufgabenstellungen haben und die Lösung weiterpflegen. Der Aufwand, einen eigenen Fork zu pflegen und nachhaltig mit dem Hauptstrang der Entwicklung zu synchronisieren, ist hoch. Leichter ist es, wenn man modular aufgebaute Software wie Python, Zope oder Plone einsetzt oder Software mit Plugin-Schnittstellen, sodass man statt eines Forks des Gesamtsystems lediglich ein individuelles Modul oder ein Plugin pflegen muss. Modulare Software bietet dem Anwender die Möglichkeit, einen Mittelweg zwischen der aufwändigen Entwicklung von reiner Individualsoftware und der passiven Nutzung generischer Open-Source-Software zu wählen. Er beginnt zwar, mit einem gewissen Aufwand ein eigenes Modul zu entwickeln, kann aber den nachfolgenden Pflegeaufwand mit der Zeit senken, wenn sein Modul von der Community angenommen wird. Das lässt sich allerdings nicht steuern, da die Akzeptanz, die ein Modul in der Community erfährt, von vielen Faktoren abhängt. Wie integriert es sich in den Gesamtzusammenhang? Löst es allgemeine Probleme auf effiziente Weise? Entspricht es den qualitativen Maßstäben der Community? Hinzu kommen kommunikative Einflüsse: Wie stark ist die Stellung des Anwenders beziehungsweise Entwicklers in der Community? Wie gut kann der Anwender beziehungsweise Entwickler sein Modul innerhalb der Community „vermarkten“? Im besten Fall wird das selbst entwickelte Modul Teil der Gesamtsoftware und der Anwender kann seinen Pflegeaufwand auf Null reduzieren. Im schlimmsten Fall muss er sich dauerhaft allein um die Pflege des Moduls kümmern. Festzuhalten bleibt jedoch, dass Open-Source-Software den Aufwand für Eigenentwicklungen senken kann, insbesondere dann, wenn die Community die individuelle Lösung annimmt und fortan unterstützt. Eine Garantie dafür kann jedoch niemand geben.

2 Weitere Einflussfaktoren werden in Stürmer und Myrach (2006) beschrieben.

3 Mit Forking bezeichnet man das Abspalten eines neuen Software-Projekts aus einem gegebenen Projekt.

Wenn der Anwender zur Entwicklung und Pflege individueller Lösungen mit einem Dienstleister zusammenarbeitet, begibt er sich in eine gewisse Abhängigkeit. Sein Geschäftserfolg ist von dem Know-how eines Dienstleisters abhängig. Will er den Dienstleister wechseln, muss dieser sich zuerst einmal in das System einarbeiten. Selbst wenn man die Abhängigkeit durch eine gute Dokumentation, die strikte Einhaltung von Standards und die Veröffentlichung des Codes in der Community reduziert, ist der Wechsel eines Dienstleisters mit Unsicherheiten verbunden.

Bei größeren Projekten kann es daher sinnvoll sein, mit mehreren Dienstleistern zusammenzuarbeiten. Dies reduziert die Abhängigkeiten und kann die Qualität der Lösungen durch eine kollaborative Entwicklung erhöhen. Allerdings entstehen Schnittstellen zwischen Auftraggeber und Dienstleister einerseits und zwischen den einzelnen Dienstleistern andererseits. Um diese Schnittstellen zu überbrücken, installiert man für gewöhnlich ein kompetentes Projektmanagement. Das Projektmanagement organisiert den Entwicklungsprozess und fungiert als Übersetzer. Es ermittelt die Bedürfnisse des Anwenders sowie den Ablauf der internen Prozesse und übersetzt diese in konkrete Spezifikationen eines Programms, die dann von den Entwicklern implementiert werden.

4 Sprints

Das Konzept der agilen Entwicklung hat den *Sprint* als besonders intensive Kollaborationsmethode hervorgebracht. Sprints ergänzen die Zusammenarbeit via Internet, sie sind zumeist mehrtägige Treffen von Entwicklern, bei denen vorher festgelegte Ziele durch eine intensive Zusammenarbeit erreicht werden. In der Python-Community können dies *PEP*s sein, in der Plone-Community *PLIP*s. In Barcellini, Détienne und Burkhardt (2006) wird beispielsweise die Implementierung von *PEP* 327 während Sprints untersucht. Sprints sind in der Plone- und Zope-Community sehr beliebt. Große Teile des CMS Plone sind bei Sprints entstanden.

Bei vielen Sprints werden aber keine formalisierten Aufgabenstellungen abgearbeitet, weil die Entwickler, die freiwillig zusammenkommen, ihren eigenen Interessen nachgehen. Besonders fruchtbar sind Sprints, um neue Softwareprojekte überhaupt erst einmal auf den Weg zu bringen. So entstanden beispielsweise die ersten Versionen von *Grok*, einem System, das das Arbeiten mit Zope drastisch vereinfacht, bei mehreren Sprints im kleinen Kreis. [4]

Sprints, bei denen *PEP*s oder *PLIP*s auf der Agenda stehen, können als anwendergetriebene Entwicklung betrachtet werden, insofern oft Wünsche der Anwender in die *PEP*s oder *PLIP*s eingeflossen sind. Bei anderen Sprints hängt die Anwenderorientierung vom Zufall beziehungsweise der Neigung der anwesenden Entwickler ab.

4 Vgl. hierzu die Blogeinträge von zwei Beteiligten in Faassen (2006), Kolman (2007) und Faassen (2007)

Der reine Anwender ist bei solchen Sprints ohnehin zumeist ausgeschlossen. Allerdings gibt es sehr unterschiedliche Arten von Anwendern, insbesondere in der Python-, Zope- und Plone-Community. Das liegt vor allem an der Art der Software. Python ist eine Programmiersprache, Zope eine Entwicklungsplattform für Webanwendungen und Plone ein *Framework* für Content-Management. Es sind *Entwickler*, die Python als Programmiersprache oder Zope als Webplattform *anwenden*. Die Einrichtung eines CMS mit Hilfe von Plone ist ab einem gewissen Anpassungsgrad ebenfalls eine *Entwickler*aufgabe. Die Fähigkeiten und Kenntnisse der entwickelnden Anwender sind naturgemäß sehr unterschiedlich, die Übergänge zwischen einem entwickelnden Anwender und einem Kernentwickler können jedoch fließend sein. In Sack et al. (2006) wird beispielhaft gezeigt, wie neue Entwickler in die Python-Community integriert werden. Ein Python-, Zope- und Plone-Anwender kann durch die Teilnahme an einem Sprint sein eigenes Know-how verbessern und vom anpassenden Anwender zu einem Entwickler von Erweiterungen werden, bis er schließlich zu den Kernentwicklern gehört.

Was aber ist mit Anwendern, die diese Hürde nicht überspringen können oder wollen, die keine Entwickler werden wollen – und dies dürfte die übergroße Mehrheit sein? Eine Lösung ist die Bildung eines Anwendernetzwerks, in dem sich Anwender mit vergleichbaren Bedarfsprofilen zusammenschließen.

5 Anwendernetzwerke

Die Anwendernetzwerke, über die im Folgenden gesprochen werden soll, sind eine eher neue Entwicklung in der Open-Source-Welt; eine Behauptung, die absurd erscheint, da die Open-Source-Communitys ja aus *Usergroups*, *Nutzer-* oder *Anwendergruppen* bestehen. Am Beispiel der Plone-Nutzergemeinschaft *PloneGov* soll deshalb dargelegt werden, was unter einem Anwendernetzwerk zu verstehen ist.

Im Jahr 2006 entstanden in Belgien und Frankreich (*CommunesPlone.org*), der Schweiz (*PloneGov.ch*) und im Baskenland (*UdalPlone*) drei Netzwerke aus Kommunen und kleinen mittelständischen IT-Unternehmen, die sich zum Ziel setzten, das Open-Source-CMS Plone mit vereinten Ressourcen an die Bedürfnisse kommunaler Verwaltungen anzupassen. Am 31. Mai und 1. Juni 2007 fand ein Sprint in Seneffe in Belgien statt, bei dem die drei kommunalen Initiativen beschlossen, sich zu einem Projekt unter dem Titel *PloneGov* zusammenzuschließen. Am 21. Juni des gleichen Jahres kam mit *Bungeni* das vierte auf Plone basierende E-Government-Projekt hinzu. *Bungeni* ist ein parlamentarisches Informationssystem, das im Rahmen des *Africa i-Parliaments Action Plan* von dem *United Nations Department of Economic and Social Affairs* (*UNDESA*) entwickelt wird. Daran sind acht nationale Parlamente aus Angola, Kamerun, Ghana, Kenia, Mozambique, Ruanda, Tanzania und Uganda beteiligt. Mittlerweile ist *PloneGov* auf 55 öffentliche Organisationen aus 15 europäischen, nord- und südamerikanischen sowie afrikanischen Ländern angewachsen.

PloneGov ist ein Netzwerk, das zu einem großen Teil außerhalb der klassischen Open-Source-Community gewachsen ist. Es bringt Nutzer zusammen, die sich nicht für Software, sondern für sehr spezielle verwaltungstechnische und kommunikative Problemlösungen interessieren. Da alle Kommunen mehr oder weniger ähnliche Anforderungen an die Software stellen, können die notwendigen technischen Lösungen für alle Mitglieder des Projekts gemeinsam und damit für die einzelne Kommune sehr viel preiswerter entwickelt werden. Zur Abstimmung dienen neben Mailinglisten vor allem Workshops, auf denen die Anforderungen formuliert werden. Anschließend werden die Erweiterungen von IT-Dienstleistern teilweise bei Sprints programmiert. In dieser Form sind bereits erste Plone-Module zur kommunalen Dokumentenverwaltung, zum Terminmanagement sowie zur Zertifizierung auf Basis der *eID card* entstanden. Die in *PloneGov* zusammengeschlossenen Kommunen, Verwaltungen und Parlamente treten der Gemeinschaft der Entwickler und den IT-Dienstleistern als starke Gruppe gegenüber. Sie befinden sich einerseits in einer bevorzugten Kundenposition, wenn sie gemeinsam Entwicklungsaufträge vergeben, andererseits bilden sie als Gruppe ein vernehmbares Sprachrohr für ihre gemeinsamen Anforderungen, die nun in der Plone-Community deutlicher vernommen werden, sodass auch hier – ohne finanzielles Engagement – Einfluss auf die Entwicklung genommen werden kann. *PloneGov* verwirklicht als eine von unten angestoßene Initiative in seiner kollaborativen Herangehensweise einige der strategischen Ziele, die die EU in ihrer *E-Government-Initiative i2010* formuliert hat. Am 13. Juni 2007 erhielt das noch junge PloneGov-Projekt den höchsten französischen Open-Source-Preis, den *Grand prix du jury des Lutèce d'Or 2007*. *PloneGov* wird von *Zea Partners* betreut, einem Zusammenschluss kleiner und mittelständischer IT-Unternehmen aus mehreren Ländern, die mit Zope und Plone arbeiten. In einer Studie der *United Nations University* über die Auswirkungen von Open Source auf den IT-Sektor wird die Kooperation *Zea Partners* seinerseits als innovatives Geschäftsmodell gewürdigt (Ghosh et al. 2006).

PloneGov ist jedoch nicht das einzige Anwendernetz in der Zope-Community. Mitte 2007 hat der *DZUG e. V.* (*Deutschsprachige Zope User Group*) die Gründung einer Hochschulgruppe initiiert. Viele Hochschulen in Deutschland, Österreich und der Schweiz nutzen – zumeist mit sehr ähnlichen Anforderungen – Zope als strategische Plattform für ihre Webanwendungen. In der Hochschulgruppe arbeiten Universitäten und Bildungseinrichtungen zusammen mit dem *DZUG e. V.* an einer Verbesserung des Informationsaustauschs. Die Hochschulen unterstützen sich gegenseitig bei dem Einsatz mittlerer und großer Installationen.

6 Die Rahmenbedingungen von PloneGov

Welche Rahmenbedingungen fördern die Bildung von Anwendernetzwerken? Wie muss die Software beschaffen sein? Was wird von den Anwendern verlangt? Und wie muss die Open-Source-Community strukturiert sein, um Anwendernetzwerken

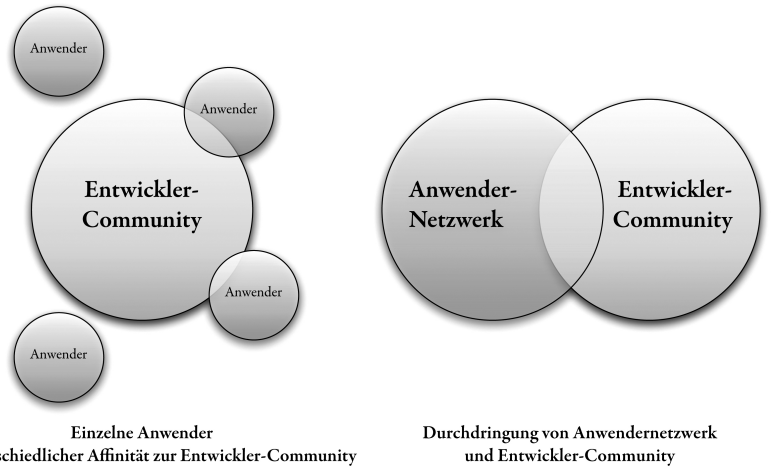

Abbildung 1: Anwendernetzwerk

ein passendes Ökosystem zu liefern? Auf diese Fragen kann man noch nicht abschließend antworten, da Anwendernetzwerke ein noch sehr junges Phänomen sind. Wir beschränken uns daher auf die Rahmenbedingungen, unter denen das Netzwerk *PloneGov* entstanden ist, um Kriterien zu finden, die die Bildung von Anwendernetzwerken begünstigen können.

6.1 Die Software

Welchen Einfluss hat die Software selbst auf die Bildung von Anwendernetzwerken? Das von *PloneGov* verwendete CMS Plone ist einerseits eine Anwendung, die auf Knopfdruck ein komplettes leistungsfähiges CMS zur Verfügung stellt. Andererseits ist es jedoch auch ein Framework, mit dem das CMS erweitert und individuellen Bedürfnissen angepasst werden kann. Dabei stützt sich Plone auf den Webapplikationsserver Zope, der wiederum eine Plattform zum Betrieb von Webanwendungen ist. Die Komponentenarchitektur von Zope sieht die Entwicklung kompakter, wiederverwendbarer Komponenten vor, die anwendungsorientiert zusammengebaut werden. Die Software ist also kein monolithischer Block, in den nur schwer neue Funktionen eingebaut werden können. Sie ist vielmehr nach allen Seiten hin offen für Erweiterungen und Alternativen. Spezielle Anforderungen können modular integriert werden, ohne dass in die Architektur der Software eingegriffen werden muss. Im Gegenteil, die Komponentenarchitektur fördert vielmehr die Integration spezieller Erweiterungen in den Gesamtzusammenhang und macht Einzelentwicklungen fast *per se* wiederverwendbar. Das ist wichtig, weil die einzelnen Kommunen und Verwaltungen trotz gleicher

Software und ähnlicher Anforderungen dennoch individuelle Konfigurationen haben. Die Anwender benötigen keine Branchenlösung, die vorgibt, alle denkbaren Probleme zu lösen und damit alle Anwender über einen Kamm schert. Vielmehr sind passende Ergänzungen für ein bestehendes modulares und individuell konfiguriertes System gewünscht. Komponentenorientierte Plattformen und Frameworks für agiles Programmieren wie Zope, Python und Plone erleichtern die Entwicklung angepasster, individueller Module mit anwenderspezifischen Funktionen. Der einzelne Anwender beziehungsweise das Anwendernetzwerk kann in kleinen überschaubaren Schritten die benötigten Module programmieren, ohne sich allzu sehr mit dem Gesamtsystem auseinandersetzen zu müssen. Auf der anderen Seite können modulare Lösungen ohne große Modifikation der Allgemeinheit zur Verfügung gestellt werden, da sie sich leicht einbinden lassen. Die Einzelentwicklung kann bei komponentenorientierter Software sehr viel problemloser als bei anders strukturierter Software einen Mehrwert für die gesamte Community erzeugen. Das setzt sich wechselseitig positiv verstärkende Prozesse der Kollaboration in Gang und schafft ein Klima, in denen Anwendernetzwerke entstehen können.

6.2 Die Anwender

Anwendernetzwerke erfordern ein hohes Maß an Eigeninitiative. Der Anwender muss nicht nur bereit und in der Lage sein, sich generell mit dem Entwicklungsprozess von Open-Source-Software auseinanderzusetzen, er muss zudem auch die Fähigkeit besitzen, unternehmens- beziehungsweise organisationsübergreifend mit anderen Nutzern zusammenzuarbeiten. Zu dieser Zusammenarbeit sind natürlich öffentliche Verwaltungen, die nicht in direkter wirtschaftlicher Konkurrenz zueinander stehen, besonders gut geeignet. Ob das Modell bei Universitäten und Hochschulen, die untereinander auch im Wettbewerb um Studenten, Professoren und Fördermittel stehen, ebenso gut funktioniert, wird die Zukunft zeigen. Es ist jedoch nicht ausgeschlossen, dass selbst konkurrierende Unternehmen gemeinsam unternehmenskritische Module und Komponenten für Zope oder Plone entwickeln und einsetzen, wenn sie eine Form der Zusammenarbeit finden, bei der ihre jeweils eigenen Geschäftsinteressen gewahrt bleiben.

Die Anwender müssen, wenn sie sich einem Anwendernetzwerk anschließen möchten, eine strategische Entscheidung für mehr Nachhaltigkeit treffen. Das bedeutet, dass sie entschlossen sein müssen, die Software und das innerhalb und außerhalb des Anwendernetzwerks erworbene Know-how langfristig zu nutzen, ansonsten lohnt der personelle und organisatorische Aufwand nicht. Ferner benötigen die Organisationen einen langen Atem, um die innovative Form der Zusammenarbeit mit allen Vor- und Nachteilen intern und extern gegen Kritik zu verteidigen. Innovationen werden, so zeigt die Erfahrung, nicht von allen Beteiligten begrüßt und von manchen sogar als Bedrohung der eigenen Position empfunden.

6.3 Das Umfeld

Eine wichtige Einflussgröße ist das Umfeld, in dem Entwickler und Anwender agieren. So ist es für die Bildung von Anwendernetzwerken erforderlich, dass sowohl die Zahl der Anwender einerseits als auch die der Entwickler andererseits eine kritische Größe überschreitet. Nur wenn es viele Anwender mit gleichen Anforderungen gibt, lohnt es sich, über die Bildung von Anwendernetzwerken nachzudenken. Und nur wenn der Kreis der Entwickler groß genug ist, stehen den Anwendern ausreichend personelle Ressourcen zur Verfügung, um ihre Anforderungen umzusetzen. Die Entwickler-Community von Plone gehört mit rund 160 Kernentwicklern und über 500 Entwicklern von Erweiterungsmodulen zu den größten Open-Source-Communitys.

Neben der Größe ist auch die Struktur einer Community bei der Bildung von Anwendernetzwerken von nicht unerheblicher Bedeutung. Welche Größe haben beispielsweise die Unternehmen, die den Einsatz der Software unterstützen? Gibt es ein oder mehrere große Unternehmen, die die Community beherrschen? Oder hat man es mit vielen kleinen und mittelständischen Unternehmen zu tun? Welche Auswirkungen hat beispielsweise die Tatsache, dass fast alle Entwickler von *OpenOffice.org* Angestellte der Firma *Sun* sind? Kann man sich hier sinnvolle Kooperationsmodelle vorstellen? Und falls es viele einzelne kleine Dienstleister gibt, sind diese gewohnt, untereinander zu kooperieren, wie es in einem Anwendernetzwerk erforderlich ist, oder schotten sie sich gegenseitig im Wettbewerb ab? In der Zope-Community ist die Kooperationsbereitschaft von Unternehmen immer schon recht groß gewesen. Das mag daran liegen, dass es von Anfang an viele kleine Zope-Dienstleister gegeben hat und auch die *Zope Corporation* selbst, die den Applikationsserver ursprünglich entwickelt hat, eher als mittelständisches Unternehmen zu bezeichnen ist. Die Gründung von *Zea Partners* im Jahre 2003 kennzeichnet jedenfalls nicht den Anfang geschäftlicher Kooperationen, sondern ist eher ein Schritt, der den bestehenden Kooperationen einen formalen Rahmen gab.

Und *last but not least* müssen auch die politischen Rahmenbedingungen stimmen. Die Legitimierung von Softwarepatenten in der Europäischen Union würde beispielsweise das Aus für Open-Source-Communitys und Anwendernetzwerke bedeuten. Neben diesem wirtschaftspolitischen Super-GAU gibt es aber insbesondere im öffentlichen Sektor auch andere kontraproduktive Momente. So können gut gemeinte Initiativen auf politischer Ebene, bei denen von oben gesteuert Softwareprojekte angestoßen werden, die Eigeninitiative der ausführenden Ebene lähmen; insbesondere dann, wenn Empfehlungen für eine bestimmte Software ausgesprochen werden oder eine Kooperation mit einem großen Hersteller oder Systemhaus eingegangen wird. *PloneGov* ist von unten gewachsen und wurde von eher kleinen Kommunen und IT-Dienstleistern gegründet.

7 Fazit

Anwendernetzwerke wie *PloneGov* sind eine viel versprechende Möglichkeit für Unternehmen, öffentliche Institutionen und Non-Profit-Organisationen finanzielles, organisatorisches und technisches Wissen zu bündeln, um gemeinsam Softwaresysteme anzupassen oder zu entwickeln, die optimal auf die professionellen Bedürfnisse der Anwender zugeschnitten sind. Dabei greifen Anwendernetzwerke sowohl auf die Leistungen und Entwicklungsprozesse der Community zurück als auch auf kommerzielle IT-Dienstleistungen. Anwendernetze kombinieren die starke Stellung des Anwenders als Kunde von individuellen IT-Dienstleistungen mit den finanziellen Vorteilen der Nutzung generischer Open-Source-Software. Anwendernetzwerke treten nach außen als Einkaufsgemeinschaft auf, die gezielt Leistungen und Know-how hinzukauft, und nach innen als Community und Selbsthilfegruppe, die ihr kollektives Know-how zum gemeinen Besten nutzt.

Literatur

Barcellini, F., Détienne, F. und Burkhardt, J.-M. (2006), Users' participation to the design process in an Open Source Software online community, *in* P. Romero, J. Good, S. Bryant und E. A. Chaparro (Hrsg.), '18th Annual Workshop on Psychology of Programming Interest Group PPIG 05'. http://hal.inria.fr/inria-00117337/en/ [11. Feb. 2008].

Barcellini, F., Détienne, F., Burkhardt, J.-M. und Sack, W. (2005), A study of online discussions in an Open-Source Software Community: Reconstructing thematic coherence and argumentation from quotation practices, *in* P. van Den Besselaar, G. de Michelis, J. Preece und C. Simone (Hrsg.), 'Communities and Technologies 2005', Springer, Berlin, S. 301–320.

Barcellini, F., Détienne, F., Burkhardt, J.-M. und Sack, W. (2006), Visualizing Roles and Design Interactions in an Open Source Software Community, *in* 'Workshop on Supporting the Social Side of large software system development at CSCW 06'.

Brand, A. und Schmid, A. (2006), Ist ein Open-Source-Projekt nur kooperativ? – Die Koordination der Zusammenarbeit im KDE-Projekt, *in* B. Lutterbeck, M. Bärwolff und R. A. Gehring (Hrsg.), 'Open Source Jahrbuch 2006 – Zwischen Softwareentwicklung und Gesellschaftsmodell', Lehmanns Media, Berlin, S. 123–137.

Faassen, M. (2006), 'Grok: or what I did on my holiday', http://faassen.n--tree.net/blog/view/weblog/2006/11/09/0 [14. Sep. 2007].

Faassen, M. (2007), 'Grok Sprint Zwei: the Ascent of Man', http://faassen.n--tree.net/blog/view/weblog/2007/01/09/0 [14. Sep. 2007].

Ghosh, R. A. et al. (2006), Study on the economic impact of open source software on innovation and the competitiveness of the Information and Communication Technologies (ICT) sector in the EU, Final report, UNU-MERIT. http://ec.europa.eu/enterprise/ict/policy/doc/2006-11-20-flossimpact.pdf [11. Feb. 2008].

Kolman, J.-W. (2007), 'Second Grok sprint - „Grokstar, the dude man"', http://jw.n--tree.net/blog/dev/python/second-grok-sprint [14. Sep. 2007].

Sack, W., Détienne, F., Ducheneaut, N., Burkhardt, J.-M., Mahendran, D. und Barcellini, F. (2006), 'A Methodological Framework for Socio-Cognitive Analyses of Collaborative Design of Open Source Software', *Computer Supported Cooperative Work (CSCW), Journal of Collaborative Computing* **15**(2-3), S. 229–250. http://www.springer.com/west/home/default?SGWID=4-40356-70-35755499-detailsPage=journaldescription [11. Feb. 2008].

Stürmer, M. und Myrach, T. (2006), Open Source Community Building, *in* B. Lutterbeck, M. Bärwolff und R. A. Gehring (Hrsg.), 'Open Source Jahrbuch 2006 – Zwischen Softwareentwicklung und Gesellschaftsmodell', Lehmanns Media, Berlin.

Architecture of Participation: Teilnehmende Open Source bei MySQL

KAJ ARNÖ

Open Source kann mehr bedeuten als die bloße Verfügbarkeit der Quelldateien. Es kann sogar umfassender sein als die Rechte, die dem Benutzer durch eine von der *Open-Source-Initiative* genehmigten Open-Source-Lizenz gewährleistet werden: die Möglichkeit, an der Entwicklung der Software teilzunehmen und zwar in gegenseitig respektvoller Zusammenarbeit mit den Hauptentwicklern. Bei MySQL wurde die aktive Teilnahme der Entwickler außerhalb der Firma *MySQL AB* lange stiefmütterlich gehandhabt. Dieser Artikel beschreibt die Bemühungen eines an sich schon erfolgreichen Open–Source-Unternehmens, die teilnehmende Entwicklung weiter aufzubauen, in Richtung des von Tim O'Reilly geprägten Begriffs *Architecture of Participation*.

Schlüsselwörter: MySQL · teilnehmende Entwicklung · Partizipation

1 Der Begriff „teilnehmende Entwicklung"

Bei der Mehrheit der Web-2.0-Unternehmen und Bewegungen wie *Wikipedia* liegt der Grund des Erfolgs in der Teilnahme der Benutzer. *Google* bietet der Welt eine Suchmaschine; die Welt stellt ihre Webseiten und ihre Suchwörter Google zur Verfügung. Wikipedia bietet der Welt die wahrscheinlich hochwertigste, umfangreichste und aktuellste Enzyklopädie; die Welt aktualisiert Wikipedia in aktiver Teilnahme und das nicht ganz ohne Eigennutzen. Jene fruchtbare Zusammenarbeit zwischen einem Unternehmen oder einer ideellen Bewegung und ihrem Umfeld bezeichnet der amerikanische Open-Source-Verleger und Industriebeobachter Tim O'Reilly als *Architecture of Participation*.

Mein skandinavischer Vorschlag zur deutschen Fassung dieses Begriffs ist „teilnehmende Entwicklung". Aus Skandinavien und Finnland kennen wir verwandte

Verhaltensweisen aus der Gesellschaft vor der Zeit des Internets. Die nordischen Länder huldigen seit langem dem *allemansrätt*, wörtlich „Jedermannsrecht". Darunter versteht man das Recht aller Menschen, das Land anderer begrenzt zu nutzen – für Zwecke wie Pilze oder Beeren sammeln, zelten oder einfach spazieren gehen. Es geht dabei nicht um ein Eigentumsrecht: Holz darf nicht gefällt, Sand oder Mineralien dürfen nicht erwirtschaftet werden und die Privatsphäre in der Nähe von Behausungen muss geschützt bleiben. Ohne die Analogie allzu weit auszudehnen, kann dies mit der kostenlosen Nutzung von MySQL unter der *GNU General Public Licence* (*GPL*) (*allemansrätt*) und der kommerziellen Nutzung von *MySQL Enterprise* (Verkauf von Holz, Sand oder Mineralien) verglichen werden. Während es beim *allemansrätt* um Benutzung und nicht so sehr um Teilnahme geht, bezieht sich der finnische Begriff *talko* auf selbstorganisierte Freiwilligenarbeit. Hierbei wird ohne den Austausch von Geld nützliche Arbeit geleistet: für Vereine, für die Gemeinde oder aber auch für einzelne Privatpersonen und Familien. All dies geschieht in der Annahme, dass die Gebenden eines Tages auch von anderen *talkos* profitieren. So können nicht nur Umzüge oder Frühjahrsputze leichter und angenehmer bewältigt werden, sondern auch heute noch gesamte Häuser oder Schiffe erbaut werden.

Die Welt des heutigen Internets ist aber nur bedingt mit sozialen Netzwerken in Skandinavien und Finnland vergleichbar, wo talko-betreibende Gruppen sich seit langem kennen, gemeinsame Werte teilen und reichlich gegenseitiges Vertrauen aufgebaut haben. Im Web 2.0 muss der Nutzen direkter sein. Beachtenswert ist aber, dass es bei der „teilnehmenden Entwicklung" nicht um eine Neuerfindung geht, sondern um eine Anpassung eines schon vorhandenen Verhaltensmusters an neue Strukturen.

2 Teilnehmende Entwicklung bei *MySQL AB* von 1995 bis 2005

Von Anfang an war MySQL Open Source, allerdings erst seit dem Jahr 2000 unter der *GPL*. Anfangs wurde MySQL hauptsächlich von einem einzigen Programmierer, Michael „Monty" Widenius, entwickelt. Das Gewicht lag aber eher auf *allemansrätt*, denn auf *talko*: Die Welt hat in den nordischen Wäldern von MySQL reichlich Pilze und Beeren gesammelt, vereinzelt wurde auch schon Holz (*MySQL Enterprise*) von den Firmengründern Monty, David und Allan gekauft. Somit konnte das Produkt weiterentwickelt und mehr Wald zum Pilze- und Beerensammeln gekauft werden.

Anfangs besaß MySQL weder Bürogebäude noch ein Domizil: Monty arbeitete in Finnland und David in Schweden. Auch heute noch arbeiten 70 Prozent der Angestellten von zu Hause aus. Als es durch Lizenz- und Supporteinnahmen mehr Geld gab, haben Monty und David, ohne Rücksicht auf die geographische Lage, Leute von den Mailinglisten angestellt. Beispielsweise arbeitete Sinisa Milivojevic vorerst in Belgrad, kurz danach in Larnaca auf Zypern, Jeremy Cole in Ohio, Indrek Siitan in Tallinn, Sergei Golubchik in Osnabrück, Tom Basil in Baltimore und so weiter. Die tägliche Kommunikation lief über *Internet Relay Chat* (*IRC*) und E-Mail. Ein paarmal

pro Jahr fand dann ein Treffen in Kalifornien, Helsinki, St. Petersburg oder Budapest statt. Auch heute noch arbeitet das Unternehmen mit über 300 Leuten in 30 Ländern sehr verteilt.

Teilnahme von außen gab es aber schon von Anfang an. Monty und David haben zunächst selbst auf Fragen in den verschiedenen MySQL-Mailinglisten geantwortet. Mit der Zeit konnten immer mehr Leute freiwillig, ohne dafür bezahlt zu werden, die Fragen Anderer sachgerecht beantworten: „Hier sind gute Pilze.“, „So bereiten Sie eine hervorragende Pilzsuppe zu.“

Die wertvollste Art der Teilnahme galt der Qualitätsverbesserung der Software. Die Anwendergemeinde setzte die Datenbank schon früh unter Bedingungen ein, die bis dato von den Entwicklern nicht getestet worden waren oder nicht getestet werden konnten. Der Wissensaustausch darüber geschah anfangs mit Hilfe einer öffentlichen Mailingliste und seit dem Jahr 2002 über eine weltweit verfügbare Fehlerdatenbank[1]: „Diese Pilze sind giftig.“ Durch diese Teilnahme unserer Community konnte das Produkt qualitativ wesentlich verbessert werden.

Auch sehr wertvoll war die Mithilfe bei der Portierung auf neue Betriebssysteme sowie beim Aufbau von APIs (*application programming interfaces*) zu anderen Programmiersprachen. Im ersten Fall ging es um Hinweise oder Ideen, die in den Programmcode einflossen, im zweiten Fall um die Entwicklung von Programmierschnittstellen, die nicht direkt zum Produkt *MySQL-Server* oder zu *MySQL AB* als Unternehmen gehören. So wurde MySQL beispielsweise auf Windows portiert und Schnittstellen zu *PHP*, *Python*, *Ruby* und vielen anderen Programmiersprachen aufgebaut: „Da gibt es Flaschen und einen Keller zum Aufbewahren der Beerensäfte.“ „Hier finden Sie Gewürze, die zur Pilzsuppe passen.“ Diese Teilnahme an der Entwicklung von MySQL hat den raschen Aufstieg des Unternehmens ermöglicht. Der Umgang mit der Community prägt die Arbeitsweise und den Alltag von MySQL–Angestellten bis heute. Trotz der vielen ehrenamtlichen Helfer blieb die Erweiterung des „MySQL-Walds“ aber immer fest in den Händen von *MySQL AB*.

3 Der Wille zur Veränderung

Mitte 2006 wurde deutlich, dass wir in puncto aktiver Teilnahme unserer Entwickler-Community zu wenig erreicht hatten. In den Jahren von 2001 bis 2006 waren so gut wie keine Beiträge in Form von Programmcode eingeflossen, was sicherlich auch an den großen künstlichen Hürden lag, die wir den Beitraggebern aufgebaut hatten. Es gab außer einer Mailingliste[2] kein gutes Kommunikationsmedium und außerdem keine Webseite mit einem Verzeichnis MySQL-kompatibler Anwendungen. Weiterhin bestand wenig Systematik im Umgang mit jenen Benutzern, die unser Produkt getestet und wertvolle Fehlerberichte zur Fehlerdatenbank hinzugefügt hatten.

1 Siehe http://bugs.mysql.com.

2 Zu erreichen über die E-Mail-Adresse internals@lists.mysql.com.

MySQL hatte viele Beitraggeber bei der externen Qualitätssicherung. Zahlreiche Menschen halfen uns indirekt, indem sie anderen MySQL-Benutzern in verschiedenen Web-Foren, Newsgroups und Mailinglisten, in vielen unterschiedlichen Sprachen, Hilfestellungen gaben. Außerdem konnte MySQL aus der Community sehr erfolgreich neue Entwickler, Support-Mitarbeiter, Berater, Schulungsleiter und sonstige Angestellte anwerben. Insofern waren wir auch in den Jahren 2001 bis 2006 erfolgreich beim Aufbau der teilnehmenden Entwicklung, allerdings noch mit reichlich ungenutztem Potenzial.

4 Die ersten Schritte nach 2006

Betrachtet man die künstlichen Hürden bei den Software-Patches, so fällt vor allem unser doppeltes Lizenzverfahren auf, wodurch *MySQL-Server* nicht nur unter der *GPL* zur Verfügung steht, sondern auch unter kommerziellen Lizenzen. Dies erfordert natürlich, dass MySQL die entsprechenden Rechte an der Software besitzt, was auch die Beiträge aus der Community einschließt. Dieses Problem wurde mit Hilfe des *Contributor License Agreement* (*CLA*) gelöst, wodurch die externen Entwickler uns die erforderlichen Rechte geben, damit deren Beiträge auch in die kommerziellen Produkte einfließen können. Jede andere Lösung hieße eine Spaltung von MySQL, nach der kommerzielle und GPL-Benutzer nicht mehr das gleiche Produkt verwenden würden, was nicht in unserem Sinne ist. Allerdings ist die Übertragung der Rechte auf MySQL ebenso im Sinne unserer Community, die sich vor allem die Mühe ersparen möchte, ihre Änderungen selbst zu verwalten und in neue MySQL-Versionen einzubauen.

Im Jahre 2006 haben wir drei weitere ausschlaggebende Schritte gemacht: Die Gründung von *MySQL Forge* mitsamt Wiki, das Starten des *MySQL Quality Contributor Programs* sowie die Einführung des *MySQL Camps*.

Das Wort *forge* in *MySQL* Forge entspricht der deutschen „Schmiede“ und ist die geläufige Bezeichnung für eine Art Verzeichnis oder Vorrat verschiedener Software, die sich mit einer gewissen Entwicklungsumgebung oder Anwendung befasst. Im Fall von *MySQL Forge* geht es dabei um Software, die mit MySQL läuft. Das können Werkzeuge wie *phpMyAdmin* sein, die die Funktionalität von MySQL leichter zugänglich machen. Es kann aber ganz generelle Software sein, die mit MySQL arbeitet. Unserer Community wollen wir dadurch die Wahl neuer Softwaretools erleichtern.

MySQL Forge ist aber nicht nur eine Sammlung von Werkzeugen, sondern auch ein Vorrat von kleinen Programmschnipseln (*snippets*), die das tägliche Leben mit MySQL erleichtern. Diese Skripte werden direkt unter *MySQL Forge* verwaltet. Darunter befindet sich z. B. *MySQL Cluster in 5 Minutes*, ein Setup-Skript für *MySQL Cluster* von Kai Voigt.

Außerdem ermöglicht *MySQL Forge* die Teilnahme unserer Community auch im Bereich der Dokumentation. Da MySQLs eigene Benutzerdokumentation aus ver-

schiedenen Gründen im schwer zugänglichen DocBook-Format zur Verfügung steht und nur von MySQL-Angestellten aktualisiert wird, können die Dokumentationen der *snippets* beziehungsweise andere Dokumentationen direkt im Wiki-Format von der Community selbst aktualisiert werden. Hier ist beispielsweise auch die oben genannte *CLA* erläutert.

Das *MySQL Quality Contributor Program* startete Ende 2006 und ist die systematische Vergütung für externe Testbeiträge. Der Beitrag der Community im Bereich von Qualitätssicherung bezieht sich in erster Linie auf Fehlerbeschreibungen, darüber hinaus aber auch auf Testfälle und bestenfalls sogar auf Fehlerbehebungen. Die „Quality Contributors" werden mit Punkten belohnt: Eine Beschreibung eines noch nicht bekannten Fehlers ist wertvoll, aber ein Testfall, der den Fehler reproduzierbar macht, ist noch wertvoller. Am wertvollsten ist natürlich ein Patch, der den Fehler behebt. Je nach Art ihres Beitrags werden auch die Quality Contributors belohnt. Die Punkte können dann nach einer gewissen Formel in Abonnements auf *MySQL Enterprise* (Basic, Silver, Gold) umgesetzt werden. Martin Friebe und Heinz Schweitzer sind die zwei fleißigsten Beitraggeber, die den Status von Gold erreicht haben. Dicht gefolgt werden sie von der Debian-User-Community.

Beim *MySQL-Camp* handelt es sich um eine Art Konferenz. Im ersten *MySQL–Camp* (November 2006) im Google-Hauptquartier in Mountain View, Kalifornien, haben sich mehr als 200 Entwickler zusammengetan und in lockerer Atmosphäre Vorträge zusammengestellt beziehungsweise diskutiert. Dabei haben die Referenten ihr Thema grob vorstrukturiert und mit dem Publikum, das eher aus Teilnehmern denn aus Zuhörern bestand, erörtert. Besonders wurden Einsatzmöglichkeiten von MySQL besprochen, wobei der Fokus oftmals auf den bestehenden Entwicklungsbedürfnissen lag. Das *MySQL-Camp* war sowohl für Benutzer als auch für Entwickler eine gute Möglichkeit angeregten Austauschs.

5 Weitere Schritte in 2007

Im Jahr 2007 wurden die meisten künstlichen Hindernisse für die teilnehmende Entwicklung der Community durch die folgenden acht Maßnahmen beseitigt. Dies heißt jedoch nicht, dass nach dem Jahr 2007 die Bemühungen eingestellt wurden, denn noch findet die Entwicklung von MySQL größtenteils durch firmeninterne Kräfte statt.

1. die Gründung des *MySQL Community Engineering Teams*
2. die öffentliche Verfügbarkeit unserer Roadmap durch *Worklog*
3. das öffentliche *Instant Messaging* (*IRC*) mit den MySQL-Entwicklern über *Freenode*
4. die öffentlich durchgeführten *Code Reviews*

5. die öffentliche Verfügbarkeit der Schulungen in *C* und *C++* der internen MySQL-Entwickler

6. die Übertragung interner Dokumentationen aus dem MySQL-Intranet zu *Forge Wiki*

7. die Teilnahme externer MySQL-Entwickler an MySQLs internem Entwicklertreffen in Heidelberg

8. das aktive Fördern neuer MySQL-Entwicklerkräfte durch den *Google Summer of Code*

Das *MySQL Community Engineering Team* wurde aus der Einsicht heraus gegründet, dass Code-Beiträge aus der Community ohne das aktive Mitwirken MySQL-eigener Entwickler kaum ihren Weg in das Produkt *MySQL-Server* finden. Vor 2007 war der Aufwand, einen Beitrag einzuarbeiten, eher eine Freizeitbeschäftigung engagierter Entwickler, deren persönlicher Erfolg daran gemessen wurde, wie viele neue Eigenschaften sie in MySQL erfolgreich programmiert beziehungsweise wie viele Fehler sie behoben hatten. Im Februar 2007 wurde eine neue Stelle geschaffen, deren Aufgabe die Motivation externer Entwickler ist. Diese sollen veranlasst werden, mehr zu machen, als unbedingt notwendig ist, um ihr eigenes Problem zu lösen. Denn eine allgemeingültige Weiterentwicklung bedarf Betriebssystemunabhängigkeit, Wartbarkeit und Zuverlässigkeit in Kombination mit anderen MySQL-Eigenschaften, die vielleicht vom ursprünglichen Entwickler gar nicht berücksichtigt wurden.

Als ein Teil von *MySQL Forge* wurde im Jahr 2007 unsere Roadmap veröffentlicht. Eine grobe Richtung war schon früher in der Dokumentation sichtbar, jedoch galt dies nicht für die Definition neuer Eigenschaften, welche nur Mitarbeitern vorbehalten war. Allerdings war dies für die wenigsten Definitionen sinnvoll, da sie zum Teil unbedingt auch von der Community kommentiert werden sollten. Das System konnte natürlich nicht ohne weiteres geöffnet werden, da *Worklog* zum Teil Informationen verwaltet, die nur von Entwicklungsleitern aktualisiert werden sollen. Wir haben uns daher für eine Informationskopie entschlossen, in der unsere Community die meisten Worklog-Einträge sehen und kommentieren kann.

Hindernisse für das Heranwachsen externer MySQL-Entwickler waren auch die fehlenden Lernmöglichkeiten. Viel lernt ein werdender Entwickler natürlich durch das Lesen des Quellcodes. Gefehlt hat es aber an Möglichkeiten, die Arbeit der internen Entwickler mitzuverfolgen, vor allem die Schritte, die vor der Veröffentlichung des Quellcodes passiert sind. Dieses Problem haben wir durch die Veröffentlichung der Instant-Messaging-Kommunikation, der Code-Reviews sowie der internen Schulungen für MySQL-Entwickler gelöst. Schon lange hat MySQL eigene IRC-Server benutzt, um in Echtzeit miteinander zu kommunizieren, beziehungsweise Probleme zu lösen. Manche dieser Diskussionen hätten genauso gut öffentlich geführt werden

können. Beides war möglich, indem wir einen neuen Kanal (#mysql-dev) unter *Freenode* eröffneten. Mitunter loggen sich die internen MySQL-Entwickler, die früher nur im hauseigenen *IRC* aktiv waren, auch unter *Freenode* ein, sprechen dort miteinander und mit unseren aktiven Beitraggebern. Die anfängliche Angst davor, mit aus Sicht der MySQL-Entwicklung belanglosen Fragen konfrontiert zu werden, war schnell vorüber. Obwohl einige Fragen, etwa zu MySQL-Fehlermeldungen unter *PHP*, in andere Foren gehören, war das Auseinanderhalten des Wesentlichen vom Unwesentlichen keine große Herausforderung. Problematischer war es, die MySQL-internen Entwickler zu ermutigen, ihre bisher internen Diskussionen nunmehr öffentlich unter *Freenode* zu führen.

Bei den Code-Reviews verhielt es sich ähnlich. Die meisten E-Mails mit Kommentaren und Verbesserungsforderungen zu Programmänderungen wurden an eine interne Mailingliste gesandt. Die strengen, aber auch hilfreichen Kommentare, die von den Teilnehmern der Reviews gegeben wurden, blieben daher zum Teil ungenutzt. Anstatt die interne Liste für Code-Reviews zu benutzen, haben wir hier auf eine weitere Liste[3], die auch externe Abonnenten hat, umgestellt. Diese werden, wenn sie später MySQL-Patches entwickeln, eine konkrete Auffassung dessen haben, was auf sie zukommen wird.

Bei der öffentlichen Verfügbarkeit der Schulungen in *C* und *C++* für interne MySQL-Entwickler ging es etwas anders zu. Die hierfür vorgesehene *MySQL University* wurde erst 2007 konzipiert und systematisiert. Von Anfang an konnte sie von der Einsicht profitieren, dass sie extern verfügbar werden sollte. Der zusätzliche Aufwand, die Schulungen offen anzubieten, hielt sich allerdings in Grenzen. Durch die virtuelle Natur der MySQL-Entwicklung (unser Motto ist *„Freedom to work anywhere @ MySQL"*) beruhen die Schulungen ohnehin auf dem Internet. Somit konnten wir uns zum Großteil auf schon vorhandene Bausteine wie *MySQL Forge Wiki* (Dokumentation) und *Freenode IRC* (Chat) beziehen. Übrig blieb nur die Frage der Audioübertragung: Wie sollte der Schulungsleiter preiswert für die ganze Welt hörbar werden? Die Entscheidung fiel auf *audio streaming* von *Ogg Vorbis*.

Eine weitere unnötige Hürde, die teilweise noch vorhanden ist, besteht in der Verfügbarkeit einer internen Dokumentation für die Entwicklungsabteilung. Diese war hauptsächlich in unserem Intranet vorhanden. Von allen Schritten, die 2007 gemacht wurden, um unsere Entwicklung zu öffnen, ist dieser der unvollständigste. Obwohl immer mehr Dokumente unter *MySQL Forge Wiki* auch von den Entwicklern und Entwicklungsleitern erstellt werden, bleiben immer noch einige Dokumente unnötigerweise „geheim". Es versteht sich von selbst, dass gewisse Termine, Pläne, Prozesse und kundenbezogene Daten sich nur intern dokumentieren lassen. Manche Dokumente könnten aber für unsere Entwickler-Community veröffentlicht werden. Die Arbeit, diese internen Dokumente auf externe Relevanz und Verfügbarkeit zu prüfen, ist noch im Gange.

3 Zu erreichen über die E-Mail-Adresse commits@lists.mysql.com.

Mit viel Vorfreude wurde die Entscheidung aufgenommen, „Externe“ zu MySQLs internem Entwicklertreffen in Heidelberg einzuladen. Unsere Erwartungen wurden jedoch noch übertroffen: Der informelle Zugang zu allen MySQL-internen Entwicklern hat zu neuen Ideen, neuen Patches und nicht zuletzt zu neuen Anstellungsverträgen geführt. Mehrere externe Community-Mitglieder, die im September unser Treffen besucht haben, arbeiten jetzt für die Firma.

Die letzte Neuerung in der Gestaltung der Zusammenarbeit zwischen MySQL und potenziellen Beitraggebern galt nicht so sehr dem Entfernen von Hürden, sondern eher dem Bau von Brücken. Wir haben das Entstehen neuer Entwicklerkräfte aktiv gefördert, indem MySQL beim *Google Summer of Code* mitgemacht hat. Von zehn genehmigten MySQL-Projekten wurden acht erfolgreich abgeschlossen. Besonders gefreut hat sich MySQL darüber, dass mit Paul McCullagh und Sheeri Kritzer zwei der Mentoren aus der Community kamen.

6 Fazit

Unsere Absicht beim Aufbau einer *Architecture of Participation* ist jene, es unserer Community zu ermöglichen, sich zu *contributors* zu entwickeln. Wir wollen es sowohl schmackhaft als auch leicht machen, zur Entwicklung von MySQL beizutragen. Zwangsläufig gibt es da Hürden und Herausforderungen, egal ob man als Entwickler innerhalb oder außerhalb der Firma arbeitet. Das Veröffentlichen des Quellcodes baut nur auf dem Grundstein der aktiven Teilnahme auf. *MySQL Forge*, *MySQL University*, der Entwicklerkanal auf *Freenode*, das *Community Engineering Team*, die öffentliche Roadmap sowie das *Quality Contributor Program* sind weitere Schritte, von denen wir uns erhoffen, dass für MySQL zum Wohl des gesamten Benutzerkreises mit der Zeit eine stetig wachsende Anzahl von aktiven, externen Beitraggebern entsteht. Dieser eingeschlagene Weg wird auch nach der Übernahme von *MySQL AB* durch *Sun Microsystems* fortgesetzt.[4]

4 Die Nachricht, dass *MySQL AB* von *Sun Microsystems* gekauft wird, kam kurz vor dem Drucktermin dieses Buchs.

OpenOffice.org – Aus dem Alltag eines nicht alltäglichen Open-Source-Projekts

JACQUELINE RAHEMIPOUR

Die freie Office-Suite *OpenOffice.org* hat nicht nur in der Open-Source–Welt einen großen Bekanntheitsgrad erreicht. Dennoch kennen nur wenige das dahinterstehende Projekt genauer. Sogar aktive Projektmitglieder haben es mitunter nicht leicht, sich in den komplexen organisatorischen Strukturen des international aufgestellten Projekts zurechtzufinden. Dieser Artikel wagt einen Blick hinter die Kulissen und durchleuchtet die verschiedenen Beweggründe, in einem Projekt wie *OpenOffice.org* aktiv mitzuwirken und es mitzugestalten. Schnell wird deutlich, dass *OpenOffice.org* einige Gemeinsamkeiten mit anderen Open-Source-Projekten aufweist, jedoch nicht nur aus historischen Gründen in vielen Bereichen alles andere als „typisch" ist.

Schlüsselwörter: Open-Source-Projekt · Community · Community-Building · Office

1 Einleitung

Die Zeiten haben sich geändert. Während *OpenOffice.org* noch vor einigen Jahren weitgehend nur in Insiderkreisen bekannt war, erfreut es sich heute allgemeiner Bekanntheit und das nicht nur im Linux-Umfeld, sondern auch bei vielen Anwendern der Windows-Plattform. Besondere Aufmerksamkeit erhält derzeit die sich noch in Entwicklung befindende Version, die nativ unter *MacOS X* lauffähig ist. Dies zeigen aktuelle Downloadzahlen, Umfragen und auch der steigende Zulauf in Anwenderforen und Mailinglisten. Immer seltener ist es notwendig, auf den einschlägigen IT-Messen im deutschsprachigen Raum zu erläutern, was sich hinter dem Namen *OpenOffice.org* verbirgt.

So positiv dies auch ist, diese Bekanntheit bezieht sich in der Regel auf das Produkt, also die freie *Office-Suite*, die sich im Grunde ständig mit dem Quasistandard messen

lassen muss. Weniger bekannt ist das Projekt *OpenOffice.org*, die Quelle, aus der alle Entwicklungen, Ideen und Strategien entspringen. Die aktiven Community-Mitglieder kennen die üblichen Vorgehensweisen, Werkzeuge sowie wichtigen Personen und finden sich im Projekt gut zurecht. Völlig anders sieht es mit den Anwendern aus, die zum ersten Mal in dieses Dickicht an ungeschriebenen Gesetzen und nur rudimentär vorhandenen Hierarchien blicken.

Tatsächlich hat die Community mit dieser Erkenntnis nicht selten zu kämpfen. Die Wahrnehmung dieser beiden Gruppen ist oft sehr unterschiedlich und stößt bei der jeweils anderen Gruppe zuweilen auf Unverständnis. So ist es nicht verwunderlich, dass die Überlegungen Außenstehender, warum ein Projekt wie *OpenOffice.org* überhaupt funktioniert und insbesondere „wer das alles bezahlt", alteingesessene Projektmitglieder eher zum Schmunzeln bringen. Andersherum scheint man als aktiver Mithelfer manchmal den Blick für die Außenwelt zu verlieren, sei es bei der Bewertung der Relevanz eines gewünschten Features oder wenn man auf ein neues Projektmitglied stößt, das sich im Dschungel des OpenOffice.org-Projekts verirrt hat.

Dieser Artikel soll etwas Licht in das Dunkel bringen und einen Blick hinter die Kulissen des Projekts *OpenOffice.org* wagen. Wie sieht der Alltag im Community-Leben tatsächlich aus? Was macht das Projekt anders als andere Open-Source-Projekte? Welche Beweggründe führen dazu, dass aus einem Anwender ein Mitstreiter wird?

2 Wie alles begann

Die Geschichte der Software *OpenOffice.org* reicht weiter zurück als die des gleichnamigen Projekts. Die freie *Office-Suite* hat ihre Wurzeln in Lüneburg, wo Marco Börries Mitte der 80er Jahre damit begann, Office-Software zu entwickeln. Über Jahre hinweg wurde *StarOffice* kommerziell entwickelt und mit Erfolg vertrieben. Im Jahr 1999 wurde die von Börries gegründete Firma *Star Division* durch *Sun Microsystems* aufgekauft, wobei auch die Office-Software und nahezu alle Entwickler übernommen wurden. Nachdem *StarOffice* im Hause *Sun* weiterentwickelt und zunächst kostenlos weitergegeben wurde, ist der Großteil des Quellcodes im Jahr 2000 offengelegt und als Open Source veröffentlicht worden.

Das Projekt *OpenOffice.org* wurde am 13.10.2000 gegründet. Seit diesem Tag haben Entwickler aus der ganzen Welt die Möglichkeit, eigene Ideen und eigenen Code in das Projekt einzubringen.

Doch Entwickler kommen nur selten von allein. Insbesondere dann nicht, wenn es sich um ein Projekt handelt, welches auf bereits vorhandenem Code basiert, der sehr umfangreich und komplex ist. Die Einstiegshürden sind in solchen Projekten erfahrungsgemäß sehr hoch. Gleichzeitig wird für eine *Office-Suite* nicht nur Programmcode benötigt. Es müssen Übersetzungen angefertigt werden, Dokumentationen wollen geschrieben sein und nicht zuletzt muss die Meinung vieler Anwender eingeholt werden, um das Produkt ständig zu verbessern.

Um mehr interessierte Helfer in das Projekt einzubinden, entstand die Idee der so genannten *Native-Lang-Projekte*. Diese Projekte kümmern sich um die Anwender und Helfer in einem bestimmten Sprachraum, übernehmen die Öffentlichkeitsarbeit, Qualitätssicherung, Dokumentation und arbeiten an der Lokalisierung der Software. Zusätzlich sind sie Anlaufpunkt für Entwickler, die zunächst Informationen über die Projektstrukturen suchen.

Nachdem das französischsprachige Projekt Mitte 2001 erfolgreich an den Start ging, wurde am 17.10.2001 das deutschsprachige Projekt gegründet. Schnell kamen Helfer hinzu, die sich in erster Linie um die Betreuung der Anwender kümmerten.

Die erste Belastungsprobe für das junge Projekt war die Veröffentlichung von *OpenOffice.org 1.0* im Mai 2002. Das Interesse der Anwender stieg sprunghaft an, die Fragen und Wünsche der Nutzer häuften sich und aus vielen begeisterten Anwendern wurden aktive Community-Mitglieder.

In den folgenden Jahren entwickelte sich das Projekt *OpenOffice.org* stetig weiter. Es entstanden verschiedene Unterprojekte, die sich speziellen Aufgabenbereichen wie z. B. der Distribution oder der Integration in *KDE* widmeten. Auch das deutschsprachige Projekt übernahm neue Aufgaben. So sind wir seit 2002 für die Kontrolle und Freigabe der deutschsprachigen Versionen zuständig und betreuen seit 2004 eine eigene CD-Distribution, die *PrOOo-Box*[1].

Ein spürbarer Wandel ergab sich in 2005 durch das Erscheinen der Version 2.0. Zum einen war diese Version auch für neue Zielgruppen interessant – insbesondere Umsteiger, die zu *OpenOffice.org* wechseln, finden sich in dieser Version besser zurecht. Zum anderen wurde aber auch deutlich, dass die vielen Helfer in der Community stolz auf ihre Arbeit und nicht immer mit dem einverstanden sind, was der Kern der Entwickler an Ideen bereithält. Seither ist ein kontinuierlicher Prozess im Gange, in dem sich Entwickler, Helfer und Anwender besser verstehen lernen. Trotz mancher Reibereien ist die Akzeptanz für den jeweils Anderen erheblich gewachsen, und aus dem Verständnis füreinander ergibt sich eine wesentlich bessere Zusammenarbeit. Als positiver Nebeneffekt mindert dieser Prozess auch die Einstiegshürden für neue Entwickler, denn Informationen werden schneller weitergeleitet und besser – wenn auch noch nicht optimal – verfügbar gemacht.

3 Organisation oder Chaos?

Anders als in einem normalen wirtschaftlich handelnden Unternehmen, in dem es mal mehr und mal weniger flache Hierarchien gibt, ist in einem Open-Source-Projekt zunächst grundsätzlich nicht klar, wer hier das Sagen hat oder wer die generelle Richtung bestimmt. Zwar gibt es für die vielen Bereiche offizielle Vertreter, die zum Teil durch aufwändige Verfahren von der Community gewählt wurden, jedoch legitimiert

1 Die Webseite der PrOOo-Box findet sich mit Liveversion und Downloadmöglichkeiten unter http://www.prooo-box.org [13. Jan. 2008].

dies keineswegs dazu, anderen Anweisungen zu geben oder eigenmächtig einen dem Community-Sinn widersprechenden Weg einzuschlagen.

Die Erfahrungen mit dem Projekt *OpenOffice.org* haben eindeutig gezeigt, dass in Communitys wie dieser die Selbstregulierung hervorragend funktioniert. Und auch wenn es bisweilen turbulent zugeht, ist das langfristige Ergebnis immer im mehrheitlichen Sinne der Community.

Die Aufgaben im Projekt *OpenOffice.org* sind so unterschiedlich und auch umfangreich, dass bereits zu Beginn der Projektarbeit verschiedene Unterprojekte ins Leben gerufen wurden, um die jeweiligen Arbeitsschwerpunkte besser abgrenzen zu können. In jedem dieser Unterprojekte gibt es einen Projektleiter (*Lead*) und mindestens einen Vertreter (*Co-Lead*), Ausnahmen bestätigen hier die Regel. Ob diese ernannt oder mittels eines Verfahrens gewählt werden, entscheiden die Unterprojekte jeweils selbst. Die Projektleiter nehmen für ihren Bereich koordinierende Aufgaben wahr, administrieren Mailinglisten und vergeben für Projektmitglieder Schreibzugriffe, z. B. für die Webseiten. Darüber hinaus sind ihre Aufgabenbereiche in der Regel nicht fest definiert – sie agieren in Abstimmung mit den anderen Projektmitgliedern. Die Unterprojekte sind in drei Kategorien eingeteilt: die *Accepted-*, *Incubator-* und *Native-Lang-Projekte*.

Unter den derzeit 28 *Accepted-Projekten*[2] sind die Unterprojekte zusammengefasst, die entweder technisch orientiert sind oder die im weitesten Sinne zur Anwenderinformation dienen. Dies sind also zum einen Unterprojekte für die unterschiedlichen Teilbereiche der Software selbst, wie Textverarbeitung (*sw*), Tabellenkalkulation (*sc*) oder Datenbank (*dba*)[3]. Zum anderen sind es Unterprojekte wie Marketing, Dokumentation oder Webseite, die für Anwender, Mithelfer und andere Interessierte wichtige Informationen bereitstellen.

Die *Incubator-Projekte*[4] sind neu ins Leben gerufene Projekte, die zunächst eine mindestens sechsmonatige Bewährungsprobe bestehen müssen, bevor sie in den Bereich der *Accepted-Projekte* wechseln können. Hier beginnen die unterschiedlichsten Unterprojekte, die die Weiterentwicklung des Projekts und der Software unterstützen sollen. Unter den aktuell 14 Unterprojekten befindet sich derzeit das Projekt *User–Experience* (*ux*), das die Schnittstelle zwischen Anwendern und Entwicklern bilden soll und die Anwenderbelange in den Vordergrund stellt. Das Projekt *Specifications* (*specs*) befasst sich mit dem Prozess der Entwicklung neuer Features und einer vollständigen Ausarbeitung dieser Funktionen, bevor ein Entwickler seine Arbeit beginnt.

Der weitaus größte Teil der Unterprojekte befindet sich in der Native-Lang-Kategorie.[5] Die Zahlen der Unterprojekte schwanken. Immer wieder kommen weitere

2 Für eine Übersicht der Accepted-Projekte siehe http://projects.openoffice.org/accepted.html [13. Jan. 2008].

3 Für jedes Unterprojekt existiert eine eigene Projektseite, die aus dem Projektnamen ableitbar ist. Das Datenbank-Projekt ist z. B. unter http://dba.openoffice.org [13. Jan. 2008] erreichbar.

4 Siehe http://projects.openoffice.org/incubator.html [13. Jan. 2008].

5 Siehe http://projects.openoffice.org/native-lang.html [13. Jan. 2008].

Sprachprojekte hinzu und es kommt auch durchaus vor, dass die Arbeit in einem sehr speziellen Sprachprojekt zeitweilig brachliegt, wenn die jeweiligen Community-Mitglieder nicht mehr die notwendige Zeit für ihre Arbeit aufbringen können. Derzeit gehen wir von ca. 100 Sprachprojekten aus, für etwa 80 davon sind die Lokalisierungen bereits zu mindestens 80 Prozent abgeschlossen. Die größten und auch sicherlich aktivsten *Native-Lang-Communitys* findet man im japanischen, französischen und deutschen Sprachraum. Von hier aus fließen sehr viele Impulse in das Gesamtprojekt mit ein.

Jedes dieser Sprachprojekte hat mit der Zeit eine eigene Struktur aufgebaut, um die dort anfallenden Aufgaben effizient bewältigen zu können. So gibt es im deutschsprachigen Projekt neben den Projektleitern weitere Ansprechpartner, die für Teilbereiche wie Qualitätssicherung, Dokumentation oder Marketing zuständig sind. Ziel ist es immer, die gebiets- und sprachabhängigen Besonderheiten zu berücksichtigen, aber trotzdem den Kontakt zu den international agierenden Projekten zu halten.

Wichtig ist dabei, dass die Einteilung der Sprachprojekte nie nach Ländergrenzen erfolgt, sondern immer die Sprache selbst im Vordergrund steht. So ist das deutschsprachige Projekt nicht nur in Deutschland aktiv, sondern auch in Österreich und dem deutschsprachigen Teil der Schweiz.

Da Englisch sowohl Kommunikationssprache im internationalen Bereich als auch „Source-Sprache" für die Programmierung ist, existiert derzeit kein englisches Sprachprojekt. Weder *OpenOffice.org* selbst noch die interne Projektkommunikation müssen also für englischsprachige Anwender übersetzt werden. Trotzdem werden die Rufe nach einem solchen Projekt immer lauter, da die Aufgaben der *Native-Lang-Projekte* heute weit über den eigentlichen Lokalisierungsprozess hinausgehen und insbesondere in den jeweiligen Sprachen auch Anwenderunterstützung, Marketingaktivitäten etc. beinhalten.

Um in diesem komplexen Gebilde von Projekten eine zentrale Anlaufstelle zu schaffen, wurde das *Community Council*[6] gegründet. Dies ist ein Gremium von neun Projektmitgliedern, die gemeinsam die Projektinteressen nach innen koordinieren und nach außen repräsentieren. Unter den Mitgliedern des *council* befinden sich Projektleiter aus den *Accepted-* und *Native-Lang-Projekten*, ein Vertreter der Community, der selbst nicht Projektleiter ist, sowie ein Vertreter der Firma *Sun Microsystems*.

4 Die Community im Tagesgeschäft

Wie in den meisten Open-Source-Projekten erfolgt die Kommunikation üblicherweise über einen der vielen Online-Wege. Zentraler Informations- und Kommunikationspunkt sind immer die Mailinglisten, auf denen alles Relevante besprochen und koordiniert wird. Die Anzahl der verfügbaren Listen ist sehr hoch, sodass es kaum möglich

6 Die Projektseite des *Community Council* findet sich unter http://council.openoffice.org/ [13. Jan. 2008].

ist, alle OpenOffice.org-Listen mitzulesen. Allein das deutschsprachige Projekt hat für die verschiedenen Teilbereiche elf verschiedene Mailinglisten[7] eingerichtet. In der letzten Zeit hat auch das Medium *IRC (internet relay chat)*[8] an Bedeutung gewonnen, um sich mit anderen Projektmitgliedern zeitgleich auszutauschen. Für bestimmte Zwecke werden IRC-Meetings abgehalten. Auf den verschiedenen *IRC*-Kanälen sind im Grunde rund um die Uhr Projektmitglieder erreichbar.

Dennoch kann nicht alles über diese Wege geklärt werden, sodass immer Wert darauf gelegt wird, auch persönliche Treffen möglich zu machen. Dazu werden zum einen die verschiedenen Messen und Veranstaltungen genutzt, auf denen *OpenOffice.org* vertreten ist, zum anderen dienen unterschiedliche interne Treffen dazu, bestimmte Projektaufgaben gemeinsam zu bewältigen. Dazu gehört im deutschen Raum ein regelmäßiges „QA-Wochenende", zu dem sich die Projektmitglieder treffen, die aktiv im Bereich Qualitätssicherung mitarbeiten.

Eine zentrale Veranstaltung auf internationaler Ebene bildet die jährliche *OpenOffice.org-Konferenz*[9], die bisher fünf Mal in verschiedenen europäischen Städten stattgefunden hat. Hier treffen sich Projektmitglieder, Entwickler und Anwender, um sich über den aktuellen Entwicklungsstand von *OpenOffice.org* zu informieren. Die zahlreichen Vorträge zeigen die vielen Facetten des Projekts, und verschiedene Meetings bringen Teams verschiedener Nationen an einen Tisch, um ohne Zeitverschiebung bestimmte Themen intensiv zu besprechen. Für die täglichen Aufgaben im Projekt werden je nach Bereich die verschiedensten Werkzeuge benötigt. Leider gibt es dafür kein „Schweizer Taschenmesser", welches für jede Anwendung nutzbar und trotzdem noch einfach zu verwenden ist. Gerade Projekteinsteiger fühlen sich schnell durch die Fülle an Werkzeugen überfordert, braucht doch jedes eine mehr oder weniger umfangreiche Einarbeitung. Am ehesten hilft es hier, sich zunächst auf eine Aufgabe zu konzentrieren und bei erfahreneren Mitgliedern nachzufragen.

Zentraler Anlaufpunkt für Informationen zu den verschiedenen Hilfsmitteln und Zugriff auf diese ist die Projekt-Webseite[10]. Neben Dokumentationen für Entwickler und Helfer erhält man hier auch Zugriff auf den *IssueTracker*[11] (die Fehler- und Featurewunsch-Datenbank des Projekts), die Quellcodeverwaltung *CVS (concurrent versions system)* oder die Mailinglistenarchive.

Ergänzt werden diese zentralen Dienste durch verschiedene, spezialisierte Online–Portale. So steht ein eigenständiges *Wiki*[12] zur schnellen Aufbereitung von Infor-

7 Unter http://de.openoffice.org/servlets/ProjectMailingListList [13. Jan. 2008] findet sich eine Übersicht über alle deutschsprachigen Mailinglisten.

8 Unter http://de.openoffice.org/about-ooo/about-irc.html [13. Jan. 2008] ist eine Aufstellung einiger dieser IRC-Kanäle verfügbar.

9 Startseite der OpenOffice.org-Konferenzen: Siehe http://marketing.openoffice.org/conference/ [13. Jan. 2008].

10 Internationale Seite: http://www.openoffice.org, deutschsprachiges Projekt: http://de.openoffice.org.

11 Unter http://qa.openoffice.org/issue_handling/pre_submission.html [13. Jan. 2008] ist der *IssueTracker* erreichbar. Hier lassen sich bereits gemeldete Fehler suchen oder neue melden.

12 OpenOffice.org-Wiki: Siehe http://wiki.services.openoffice.org/wiki/Main_Page [13. Jan. 2008].

mationen bereit. Detaillierte Informationen über die Integration von Quellcode in die aktuellen Entwickler- oder Release-Versionen erhält man im *EIS (Environment Information System)*[13]. Für Übersetzungen wird daran gearbeitet, *Pootle-Server*[14] zur Verfügung zu stellen, um Lokalisierungen online bearbeiten zu können. Weitere Dienste helfen dabei, den Freigabeprozess für neue Versionen zu überwachen, die Verteilung der fertigen Dateien im *Mirror-Netzwerk* zu kontrollieren oder das Erstellen von Entwicklerversionen auf entfernten Rechnern zu ermöglichen. Gerade Letzteres – das Erzeugen oder „Bauen" einer OpenOffice.org-Version – wird aber nach wie vor eher lokal auf dem Rechner des Entwicklers stattfinden.

5 Unterstützung des Projekts durch Unternehmen

So vielfältig die Aufgaben im OpenOffice.org-Projekt sind, so vielschichtig ist auch die Zusammensetzung des Projekts. Den zahlenmäßig größten Anteil am Projekt stellen dabei tausende freiwillige Helfer, die oft während der Freizeit Beiträge zu *OpenOffice.org* leisten. Zusätzlich zu diesen privaten Beteiligten zählen aber auch verschiedene Firmen zum Projekt. Die Palette dieser Firmen reicht vom Einzelunternehmen – möglicherweise ein lokaler Schulungsanbieter – bis hin zu den „Schwergewichten" der IT-Branche, wie *Sun Microsystems* oder *IBM*. Gerade diese Mischung aus Firmen und freien Beteiligten unterscheidet *OpenOffice.org* von den meisten Open-Source-Projekten.

Firmenbeteiligungen sind für das Projekt vielleicht nicht lebensnotwendig, aber auf jeden Fall sehr wertvoll. Dadurch, dass viele der beteiligten Unternehmen Mitarbeiter direkt für die Arbeit an *OpenOffice.org* beschäftigen, wird eine höhere Stabilität und auch Planbarkeit in der Entwicklung der Software erreicht. Festangestellte Programmierer können wesentlich mehr Zeit für das Projekt aufwenden und in der Regel auch über einen längeren Zeitraum hinweg an einer speziellen Aufgabe arbeiten.

Neben personellen Kapazitäten werden von den Unternehmen auch Werkzeuge und Ressourcen bereitgestellt, die für die Community nur schwer zu organisieren wären. So betreibt *Sun Microsystems* eine Build-Umgebung, in der regelmäßig Entwicklerversionen für verschiedene Plattformen und Sprachen erstellt werden. Im Anschluss daran werden diese Versionen innerhalb von *Sun* auf verschiedenen Rechnern automatisch getestet. Könnte man hierbei nicht ein lokales Netz nutzen, wäre bereits die Verteilung der anfallenden Datenmengen ein Problem.

Bei allen positiven Effekten, die eine solche Firmenbeteiligung für das Projekt hat, gibt es natürlich auch Schwierigkeiten. So vergisst man schnell, dass es auch eine „Welt da draußen" gibt, wenn der nächste Entwickler im Büro nebenan ist. Hier hilft es nur, Disziplin zu halten und die wichtigsten Arbeiten und Absprachen zu dokumentieren.

13 Environment Information System: Siehe http://eis.services.openoffice.org/EIS2/ [13. Jan. 2008].

14 Für weitere Informationen zur freien Übersetzungssoftware *Pootle* siehe http://translate.sourceforge.net/wiki/pootle [07. Feb. 2008].

Gleiches gilt für Werkzeuge, die innerhalb einer lokalen Umgebung entwickelt werden. Den größten Nutzen für die Community haben diese Werkzeuge dann, wenn sie von anderen eingesetzt oder adaptiert werden können. In den letzten Jahren wird deshalb ständig versucht, diese Werkzeuge zu dokumentieren bzw. direkt unter einer Open-Source-Lizenz zu veröffentlichen.

Die Mitarbeit von Firmen innerhalb der OpenOffice.org-Community hat eine lange Geschichte. *Sun Microsystems* als Gründer des Projekts stellt von Beginn an den Großteil der Entwickler und trägt auch einen beachtlichen Anteil zur Qualitätssicherung, Lokalisierung und vielen weiteren Teilbereichen bei. Nach wie vor stammen etwa 90 Prozent der Quellcode-Beiträge von *Sun*.

Schon frühzeitig arbeiteten aber auch verschiedene Linux-Distributoren im Projekt mit. *Red Hat* steuert z. B. kontinuierlich Fehlerkorrekturen und Verbesserungen bei. *Ximian* arbeitete schon während *OpenOffice.org 1.0* an einer Integration mit *Evolution* und stellte eine alternative Build-Umgebung bereit. *SuSE* konnte bereits in *OpenOffice.org 1.1* eine Integration mit dem *KDE-Desktop* vorzeigen und stellte entsprechende *icon sets* oder den Zugriff auf die typischen *KDE* Datei- und Druckdialoge bereit. Die Entwickler dieser beiden letztgenannten Firmen arbeiten heute zusammen mit weiteren unter dem Dach von *Novell* und kümmern sich dort unter anderem um die Erhöhung der Performance, Verbesserungen in Teilfunktionen von *Calc* oder auch die Verarbeitung von *Microsoft Office VBA*-Code in *OpenOffice.org*. Relativ neu im Projekt sind die Firmen *Red Flag Software Co.,Ltd* aus Beijing und *IBM*.

Red Flag besitzt seit einigen Jahren Erfahrungen mit einer eigenen Version von *OpenOffice.org* auf dem chinesischen Markt und sieht sich dort mit einer vollkommen anderen Anwenderkultur konfrontiert als die meisten westlich orientierten Entwickler im Projekt. Dieses Wissen und die Anpassungen am Produkt sollen in nächster Zeit direkt in *OpenOffice.org* zurückfließen. Auch *IBM* vertreibt ein eigenes Produkt, welches auf *OpenOffice.org* basiert. Insbesondere die Zugänglichkeitstechnologien, die für *Lotus Notes* bzw. *Lotus Symphony* geschaffen wurden, sollen *OpenOffice.org* zur Verfügung gestellt werden. Mit der großen Erfahrung rund um das Dateiformat *OpenDocument* ist *IBM* ein idealer Partner für die Community.

Neben diesen Großfirmen gibt es aber auch viele kleinere Firmen und Institutionen, die direkt zu *OpenOffice.org* beitragen und nur wenig bekannt sind. So stammt die Unterstützung für *Smart Tags* in *Writer* zum Großteil von der österreichischen Firma *Fabalabs Software GmbH*. Die seit Version 2.0.2 integrierte Rechtschreibprüfung *Hunspell* wurde wesentlich durch die *Technische Universität Budapest* unterstützt. Die Verbesserung der Autosummen-Funktion in *OpenOffice.org 2.3* ist hauptsächlich der französischen Firma *Linagora* zu verdanken. Genauso wichtig sind aber auch kontinuierliche Rückmeldungen aus Eigenentwicklungen rund um *OpenOffice.org*, wie sie z. B. durch das *LiMux*-Team der Stadt München[15] erfolgen.

15 Vgl. Hoegner (2006).

6 OpenOffice.org und die Außenwelt

Auch die beste Software der Welt nutzt wenig, wenn sie niemand kennt. Wer Feedback erhalten will, muss offen auf die Anwender zugehen und durch kontinuierliche Präsenz Interesse schaffen. Nicht nur die Offenheit des Quellcodes, sondern auch die Offenheit der Beteiligten ist ein entscheidender Vorteil für freie Softwareprojekte. Ein wichtiges Anliegen für uns ist es, diese Offenheit zu demonstrieren. Nichts ist dazu besser geeignet, als das persönliche Gespräch z. B. auf Messen oder Open-Source–Tagungen.

Die hierfür zur Verfügung stehenden Mittel sind üblicherweise sehr begrenzt und so muteten manche Messeauftritte etwas befremdlich an, wenn das gesamte Messeequipment aus einem Stuhl, einem Tisch, einem darauf befindlichen Laptop und einem mehr schlecht als recht gestalteten Plakat bestand. Das war Alltag in den Anfängen der meisten Open-Source-Projekte, die den Weg in die Öffentlichkeit angetreten haben.

In diesem Punkt hat sich *OpenOffice.org* stark „gemausert". Auch wenn hierbei hauptsächlich die Erfahrungen aus dem deutschsprachigen Raum angeführt werden können, muss sich das Projekt *OpenOffice.org* sicherlich nicht verstecken.

Heute ist *OpenOffice.org* auf allen wichtigen IT-Veranstaltungen im deutschsprachigen Raum vertreten und ist hier Ansprechpartner für Anwender und Interessierte, die es vielleicht werden wollen.[16] Häufig werden im Rahmen der Veranstaltungen Vorträge oder Workshops angeboten, und auch direkt am Messestand wird bereitwillig Auskunft über den aktuellen Stand bzw. zukünftige Entwicklungen gegeben. Nicht selten kommt es vor, dass Anwender gezielt den Weg zu den Messeständen suchen, um konkrete Fragen zur Anwendung zu stellen oder sogar Beispieldokumente mitbringen.

Auf größeren Messen wie der *CeBIT* in Hannover oder der *Systems* in München tritt *OpenOffice.org* häufig an Gemeinschaftsständen auf. Hier präsentieren verschiedene Firmen, die im *OpenOffice.org*-Umfeld tätig sind, ihre Produkte und Dienstleistungen und zeigen damit den Besuchern, dass *OpenOffice.org* keineswegs nur für den Privatanwender, sondern auch im Unternehmenseinsatz geeignet ist.

Solche Gemeinschaftsstände und auch weitere Marketingaktivitäten werden über den Verein *OpenOffice.org* Deutschland e. V.[17] organisiert und (vor)finanziert. Diese rechtliche Struktur gibt es seit 2004 und unterstützt die Projektmitglieder bei ihrer Arbeit. Mittlerweile können aus den verfügbaren Spendengeldern auch kleinere Entwicklungsprojekte und vom Projekt organisierte Veranstaltungen gefördert werden.

16 Unter http://de.openoffice.org/veranstaltungen/index.html [13. Jan. 2008] findet sich eine Veranstaltungsübersicht des deutschsprachigen Projekts.

17 Siehe die Webseite des Vereins OpenOffice.org Deutschland e. V. unter http://www.ooodev.org [13. Jan. 2008].

7 Community-Building bei OpenOffice.org

Das auf Messeauftritte, eigene Veranstaltungen und Medienpräsenz bauende Produktmarketing ist jedoch nur ein Aufgabenbereich. Mindestens genauso wichtig ist es für uns, auch aktiv Projektmarketing zu betreiben, um so auch neue Mithelfer zu interessieren.

Die Erfahrung hat aber gezeigt, dass es für einen Außenstehenden sehr schwer zu erkennen ist, wo er sich einbringen kann. Besonders überraschend war die Erkenntnis, dass das Projekt nach außen häufig wie eine geschlossene Gesellschaft wirkt. Potenzielle Mithelfer fragen sogar an, welche Bedingungen erfüllt werden müssen, um überhaupt mithelfen zu „dürfen". Selbst schriftliche Bewerbungen sind bereits eingegangen.

Um in diesem Bereich aufzuklären und insbesondere potenziellen Mithelfern die erste Hürde zum Projekt zu nehmen, organisiert das deutschsprachige Projekt eigene Veranstaltungen und informiert dort über die Community-Arbeit. So soll das *OOoCamp* als regelmäßiges Event etabliert werden, welches neue Projektmitglieder in die typischen Arbeitsbereiche bei *OpenOffice.org* einführt und damit den Einstieg erleichtert. Es ist kaum verwunderlich, dass im Grunde alle Open-Source-Projekte um weitere Mithelfer ringen. Es gibt immer genug zu tun und durch die Fülle der anfallenden Arbeiten nimmt die Koordination der Projektmitglieder und der Aufgaben immer mehr Zeit in Anspruch.

Bei all den vielfältigen Aufgaben ist im Grunde für jedes Community-Mitglied das Passende dabei. Nicht nur in den technischen Bereichen Entwicklung oder Qualitätssicherung stehen noch viele weitere Arbeiten an, sondern auch in leichter zugänglichen Bereichen wie Dokumentation, Übersetzung oder Marketing.

Dennoch braucht die Community für die meisten Aufgaben Anleitung. Zentrale Personen müssen die Fäden zusammenhalten und die Mithelfer koordinieren. Dabei steht die inhaltliche Betreuung weniger im Vordergrund, vielmehr benötigen gerade die neuen Projektmitglieder eine direkte Anlaufstelle – und noch viel wichtiger: Sie benötigen Rückmeldungen. Dieser Aspekt wird häufig unterschätzt, nur wenige Projektmitglieder schöpfen ihre Motivation zur Mitarbeit aus der Arbeit an sich.

7.1 Wege zu OpenOffice.org

Genaue Werte über die Anzahl der Community-Mitglieder existieren nicht. Ebensowenig gibt es eine Definition, was ein aktives Community-Mitglied auszeichnet. Konkrete Zahlen lassen sich bestenfalls über die eingeschriebenen Personen auf den jeweiligen Projektmailinglisten und über die angemeldeten Benutzer in den Unterprojekten ableiten. Diese sagen aber nichts über die Intensität der Mitarbeit aus. In Schätzungen wird häufig von ca. 30 000 aktiven Mithelfern weltweit gesprochen. Betrachtet man das deutschsprachige Projekt und die Projektmitglieder, die aufgrund ihrer Berechtigungen einen höheren Status erlangt haben, kann man im *de-Projekt*

von ca. 300 Community-Mitgliedern ausgehen. Die Zahlen der eingeschriebenen Personen auf den Mailinglisten liegen jedoch weit höher. Trotzdem wird die regelmäßig zu erledigende Arbeit von einer deutlich kleineren Gruppe übernommen, die schätzungsweise bei ca. 50 Personen liegt. Hinzu kommen weitere Personen, die einzelne Aufgaben übernehmen.

Die Wege in das Projekt *OpenOffice.org* hinein sind so unterschiedlich wie die Personen, die man dort antrifft. Ein häufiger Einstiegspunkt ist die Anwenderunterstützung, die vom Projekt über Mailinglisten angeboten wird. Hier finden sich Anwender mit speziellen Fragestellungen ein und sind häufig überrascht, wie schnell und unkompliziert ihnen geholfen wird. Insbesondere Business-Anwender oder Studenten, die beispielsweise an ihrer Abschlussarbeit arbeiten, nutzen dieses Angebot regelmäßig. Eher schleichend werden einige dieser Fragesteller dann zu Antwortgebern, um anderen diese Hilfe anzubieten, die sie selbst erfahren haben.

Viele Community-Mitglieder haben auch direkten beruflichen Hintergrund. Sie sind beispielsweise Lehrer, Trainer oder bieten Dienstleistungen im Zusammenhang mit *OpenOffice.org* an. Ihre Mitarbeit begründet sich häufig aus dem Wunsch, dem Projekt „etwas zurückzugeben". Die Impulse, die von diesen Personen in das Projekt gelangen, sind sehr wertvoll, da sie Erfahrungen von Anwendern einbringen können, die ansonsten nie weitergegeben würden. Im Gegenzug ist das Projekt eine perfekte Informationsquelle für den intensiven Umgang mit dem Programm. Für einige Community-Mitglieder stehen die Faktoren Bildung und Wissensaufbau im Vordergrund. So trifft man häufig Studienanfänger an, die im Rahmen ihrer Interessen Hilfe zur Verfügung stellen. Arbeitssuchende wollen durch ihre Mitarbeit einen Teil ihrer Freizeit mit sinnvoller Beschäftigung ausfüllen. Heutzutage bedeutet die Community-Arbeit in verschiedenen Bereichen durchaus eine Weiterbildung, die sich auf den beruflichen Lebensweg positiv auswirken kann. Nicht nur die rein inhaltliche Qualifikation ist hier entscheidend, sondern insbesondere die Aneignung von *Soft skills* wie Teamfähigkeit, selbständiges Arbeiten und die Übernahme von Verantwortung, die sich nicht auf vorhandene Hierarchien stützt, sondern auf der eigenen Überzeugung aufbaut.

Ein wichtiger Grund zur Mitarbeit ist auch häufig der Wunsch, die Entwicklung des Programms mitzugestalten und selbst dazu beizutragen, welche Veränderungen die zukünftigen Versionen erfahren sollen. Darauf können nicht nur Entwickler Einfluss nehmen, sondern insbesondere auch Mithelfer der Qualitätssicherung, Autoren von Spezifikationen, Mithelfer im *User-Experience-Project* und sogar jeder Anwender, der im Voting-System des *IssueTracker* seine Stimme für Features f abgibt, die ihm besonders wichtig sind.

Unbestritten ist der Einstieg für Entwickler deutlich schwerer. Während in anderen Bereichen auch häufig kleine und überschaubare Aufgaben anstehen, verpflichtet sich ein Entwickler in der Regel zwangsläufig für einen größeren Zeitraum zur Mitarbeit. Die Einarbeitungszeit ist erfahrungsgemäß lang, bis die ersten Resultate der eigenen Arbeit sichtbar werden. Für neue Entwickler ist insbesondere der nahe Kontakt zu

erfahrenen Entwicklern wichtig, der in der Regel über Mailinglisten oder über die IRC–Kanäle am einfachsten ist. Auch Projekte wie der *Google Summer of Code* können helfen, interessierten Entwicklern den Einstieg bei *OpenOffice.org* zu erleichtern.

7.2 Die Motivation zur Mitarbeit

In einem Open-Source-Projekt mitzuarbeiten, ist sicherlich häufig eine Frage von Anerkennung durch die Community und der Bestätigung der eigenen Leistung. Es gibt ein gutes Gefühl, jemandem mit einer fundierten Antwort zu helfen und dafür ein Dankeschön zu erhalten. Ob dies auf einer Mailingliste stattfindet oder im direkten Gespräch auf einer Messe, ist dabei eher nebensächlich.

Bei der Mitarbeit direkt am Programm – sei es als Entwickler, Übersetzer oder Tester – ist häufig das Wissen, etwas Greifbares beigetragen zu haben, schon ausreichend. Das eigene Feature, die fertige Dokumentation oder Übersetzung im fertigen Programm zu sehen, ist für viele bereits Motivation genug.

Dennoch ist es gerade für Außenstehende nur schwer verständlich, wieso man seine Freizeit für die Weiterentwicklung eines Office-Pakets aufwendet und nicht etwa andere gemeinnützige Arbeit leistet. Um dieser Frage etwas tiefer auf den Grund zu gehen, habe ich im Frühjahr 2007 als Vorbereitung für einen Vortrag zu *OpenOffice.org* eine kurze Umfrage im deutschsprachigen Projekt gestartet. Die Fragestellungen sollten vor allem klären, warum die Community-Mitglieder bei *OpenOffice.org* mitarbeiten und wie sie auf das Projekt aufmerksam wurden.

Diese Umfrage war keineswegs repräsentativ, dennoch zeigen die Antworten der ca. 50 Beteiligten eine gute Tendenz und sie stimmen durchaus mit persönlichen Erfahrungen im Projekt überein.

Die Auswertung der Altersstruktur zeigte, dass ein Großteil der Befragten im Alter zwischen 40 und 60 Jahren alt ist. Wahrscheinlich liegt das tatsächliche Durchschnittsalter im Projekt um einige Jahre niedriger, jedoch ist unumstritten, dass nur wenige sehr junge Mithelfer den Weg zu *OpenOffice.org* finden und dauerhaft dabei bleiben.

Wenig überraschend war der Anteil der männlichen Befragten (ca. 94 Prozent) und der Anteil der Community-Mitglieder, die mit *OpenOffice.org* auch beruflich zu tun haben (ca. 54 %).

Zur Frage, wie die Befragten auf das Projekt *OpenOffice.org* aufmerksam wurden, sind sehr häufig persönliche Empfehlungen genannt worden („Beschwerde eines Linux-Users über ein von mir zugesandtes Word-Dokument“ oder „Hinweis eines Kommilitonen, der mein Gejammer über Microsoft Office nicht mehr ertragen konnte“). Zudem erfolgt die Bekanntmachung mit dem Projekt häufig über Artikel in Zeitschriften oder über CD-Beilagen.

Eine weitere Frage sollte ergründen, warum die Befragten bei *OpenOffice.org* mitarbeiten. Die Antworten spiegeln stark die zuvor geschilderten Gründe wider: Im Vordergrund stehen hier der Lernfaktor („Selber fragen, anderen antworten“), das Erbringen einer Gegenleistung („Habe in jeder Hinsicht viel von Open-Source-Soft-

ware anderer Autoren profitiert und möchte mich revanchieren"), das Gruppengefühl („Mich inspiriert die Gemeinschaft, die etwas erschafft ohne kommerzielle Interessen im Vordergrund") und die Unterstützung des Open-Source-Gedankens („Weil es richtig und wichtig ist, dass Software frei ist, und ich dafür einen winzigen Baustein beitragen kann").

7.3 Der Frauenanteil bei OpenOffice.org

Diesem Thema widme ich einen eigenen Abschnitt – obwohl es im Grunde kein Thema ist. Unbestritten ist die Frauenquote in Open-Source-Projekten gering. Verlässliche Zahlen lassen sich nur schwer ermitteln, jedoch liegt der Frauenanteil im deutschsprachigen OpenOffice.org-Projekt schätzungsweise bei 5-10 %, gemessen an den Personen, die sich auf der deutschsprachigen Projektmailingliste in die aktuellen Geschehnisse einbringen. Zu wenig, wie viele finden. Genau betrachtet ist das aber keine Besonderheit von Open-Source-Projekten, sondern im IT-Umfeld durchaus üblich.[18]

Da eine Office-Software aber auch von weniger technisch interessierten Anwendern benutzt wird, als viele andere Programme, ist der Frauenanteil in diesem Projekt möglicherweise sogar höher als in anderen. Ähnlich verhält es sich mit dem Anteil von z. B. Ärzten oder Anwälten, die aktiv in der OpenOffice.org-Community mithelfen.

Insofern ist bei *OpenOffice.org* kein Ungleichgewicht oder gar eine Diskriminierung erkennbar. Im Gegenteil: Bemerkenswert ist die Tatsache, dass viele der weiblichen Community-Mitglieder zu den aktivsten und zuverlässigsten gehören. Es scheint, als hätten sich diese Frauen ihre Entscheidung gründlicher überlegt und sind anschließend auch nachhaltiger dabei. Ohne Zweifel war es für viele eine Überraschung, als im Jahre 2006 zumindest für kurze Zeit gleich zwei Frauen an der Spitze der *de-Community* standen.

8 Fazit

OpenOffice.org ist den Kinderschuhen entwachsen und führt längst kein Nischendasein mehr. Nicht nur die stetig steigenden Anwenderzahlen, sowohl im privaten als auch im geschäftlichen Umfeld, zeigen eindrucksvoll die Beliebtheit des Office-Pakets.

Während das Produkt *OpenOffice.org* bereits große Bekanntheit erlangt hat, bleibt das Projekt *OpenOffice.org* häufig im Verborgenen. Klassisches Marketing greift hier nicht, wenn es um die Suche nach aktiven Projektmitgliedern geht. Die Komplexität des Projekts ist hierbei Hürde und Herausforderung zugleich, macht aber für viele gerade den besonderen Reiz aus.

Die Beweggründe, in einem Open-Source-Projekt wie *OpenOffice.org* mitzuarbeiten, lassen sich sicherlich nicht verallgemeinern. Viel zu sehr spielen persönliche

18 Vgl. Jung (2006).

Erfahrungen, das eigene Umfeld und nicht zuletzt sogar der Zufall eine Rolle. Jedes Community-Mitglied schreibt seine eigene Geschichte und stellt mit seiner Mitarbeit und seinen Impulsen einen wichtigen Baustein des Ganzen dar.

Schon historisch bedingt partizipieren jedoch neben vielen Einzelpersonen auch Firmen und Organisationen an der Weiterentwicklung des Projekts. Diese symbiotische Verbindung gereicht trotz aller Schwierigkeiten zu beiderseitigem Vorteil. So ergeben sich mit der Fülle der teilnehmenden Partner die verschiedensten Blickwinkel und Anforderungen, die in ihrer umfassenden Dynamik die zukünftigen Ziele und Richtungen beeinflussen. Nicht zuletzt ist gerade das Mitwirken von Firmen einer der Gründe für Kontinuität und Planbarkeit von Projekt und Programm.

Historie und Projektbeteiligte machen *OpenOffice.org* zu keinem „typischen" Open-Source-Projekt. Die vielfältigen Einflüsse und die Dynamik der Community lassen die Zukunft stets spannend bleiben.

Literatur

Hoegner, W. (2006), Das Projekt LiMux – Freie Software für die Münchner Verwaltungsclients, *in* B. Lutterbeck, M. Bärwolff und R. A. Gehring (Hrsg.), 'Open Source Jahrbuch 2006 – Zwischen Softwareentwicklung und Gesellschaftsmodell', Lehmanns Media, Berlin, S. 59–72.

Jung, P. (2006), Frauen-freie Zone Open Source?, *in* B. Lutterbeck, M. Bärwolff und R. A. Gehring (Hrsg.), 'Open Source Jahrbuch 2006 – Zwischen Softwareentwicklung und Gesellschaftsmodell', Lehmanns Media, Berlin, S. 235–250.

F

OpenMoko – Freie Software für Mobiltelefone

MICHAEL LAUER

OpenMoko ist eine freie Softwareplattform für Mobilkommunikation, die derzeit unter der Leitung von *OpenMoko, Inc.* entwickelt wird. Finanziert wird das Projekt von *First International Computer, Inc.*, die sich auch für die zugehörige freie Mobiltelefonie-Hardware verantwortlich zeichnen. Der folgende Artikel beschreibt die bisherigen Schritte und Erfolge hin zu einem völlig auf offener Software basierendem Mobiltelefon. Weiter wird ein Vergleich mit ähnlichen Projekten angestellt und ein Ausblick in die Zukunft quelloffener Mobiltelefonieplattformen gegeben.

Schlüsselwörter: Offene Mobiltelefonieplattform · Ubiquitous Computing · Offene Hardware

1 Software auf Mobiltelefonen

Im Gegensatz zur offenen Softwarewelt auf PCs, Laptops und ähnlichen Geräten findet man im Umfeld von Mobiltelefonen ein geschlossenes System. Die Zusammensetzung der Betriebssoftware wird im Wesentlichen von Mobilfunkbetreibern kontrolliert (*Branding*) und schränkt die möglichen Anwendungsszenarien dadurch erheblich ein. Ein prominentes, abschreckendes Beispiel hierfür ist, dass ein Mobilfunkbetreiber von einem Hersteller forderte, ein Bluetooth-Profil zur Dateiübertragung zu entfernen – um so die Benutzer auf (kostenpflichtig) über GSM-Datendienste zu beziehende Zusatzsoftware und -medien festzulegen.

Für Entwickler ist die Situation ähnlich desolat. Das Ausführen eigener Programme auf Mobiltelefonen ist bei gängigen Produkten im Wesentlichen auf JAVA-Anwendungen limitiert, die in einer kontrollierten Laufzeitumgebung von den übrigen Subsystemen abgetrennt werden. Dies bedeutet, dass weder beliebige Programmiersprachen noch Bibliotheken verwendet werden können. Ebenso ist es nicht möglich, das komplette Betriebssystem des Mobiltelefons auszutauschen – etwa um mit alternativen

Betriebssystemen zu experimentieren oder anwendungsspezifische Optimierungen vorzunehmen.

Mag man dies bei Mobiltelefonen, die proprietäre Plattformen einsetzen, noch akzeptieren, so ist es umso unbefriedigender, dass diese Beschränkungen bis dato auch auf allen Linux-basierten Plattformen vorliegen. So basiert beispielsweise die weit verbreitete EZX-Plattform[1] von *Motorola* auf einem Linux-Kern – alle weiteren Software-Komponenten sind jedoch proprietär und verhindern die Ausführung von nativen (d. h. nicht in einer virtuellen Maschine gekapselten) Anwendungen.

Der Einsatz beliebiger freier Software auf Mobiltelefonen wird hiermit nahezu unmöglich gemacht – der erschwerte Zugang für Entwickler mündet in einem Ausschluss von Talenten und in Konsequenz auch von neuen Ideen, Anwendungen und Interaktionsformen.

2 Die Softwareplattform OpenMoko

2.1 Ziel

Das Projekt *OpenMoko*[2] hat sich zum Ziel gesetzt, eine vollständige – ausschliesslich aus freier Software bestehende – Plattform für mobile Kommunikationsgeräte zu entwickeln. Mobiltelefone sind hierbei nur eine mögliche Ausprägung, aufgrund ihrer Allgegenwärtigkeit zunächst jedoch das primäre Ziel. *OpenMoko* wurde von dem Entwickler Sean Moss-Pultz ins Leben gerufen, der hiermit den ersten ernstzunehmenden Ansatz für eine komplett offene Plattform gewagt hat. Um dabei nicht nur technologisch, sondern auch legal auf der sicheren Seite zu sein, wurden gleich zu Beginn Schlüsselpersonen aus der freien Entwicklergemeinschaft in das Kernteam eingebettet. Finanziert und unterstützt wird das Projekt von *OpenMoko, Inc.* – einer Ausgründung des taiwanesischen Elektronikkonzerns *First International Computer*[3] (FIC), der hier auch als Hardwarepartner auftritt und *OpenMoko* in künftigen Produkten einsetzen wird. Den Anfang hat FIC bereits mit dem für *Dash Navigation, Inc.*[4] hergestellten Navigationsgerät *Dash Express* gemacht.

2.2 Softwarearchitektur

Abbildung 1 zeigt die Softwarearchitektur der Plattform wie sie zu Beginn des Projekts (Ende 2006) entworfen wurde. *OpenMoko* setzt auch in der gegenwärtigen Version noch hauptsächlich Bibliotheken des GNOME-Projekts[5] ein – zukünftige Arbeiten werden jedoch auch Entwicklungen des Enlightenment-Projekts[6] integrieren, deren

1 *Motorola A780, A1200, Rokr E2, Rokr E6, Razr 2*, u. v. m.
2 Siehe http://www.openmoko.org bzw. http://www.openmoko.com.
3 Siehe http://www.fic.com.tw.
4 Siehe http://www.dash.net.
5 Siehe http://www.gnome.org.
6 Siehe http://www.enlightenment.org.

Stärke die effiziente Realisierung graphisch ansprechender, dynamischer Benutzungsoberflächen ist.

Zum jetzigen Zeitpunkt (Januar 2008) ist der Reifegrad der OpenMoko-Plattform als Betaversion einzuschätzen. Sowohl der Umfang als auch die Stabilität müssen für einen Endkundenbetrieb noch verbessert werden. *OpenMoko, Inc.* selbst konzentriert sich derzeit hauptsächlich auf die Erstellung der Basisfunktionalität, insbesondere der Middleware, um somit als Katalysator für Anwendungen Dritter zu dienen.

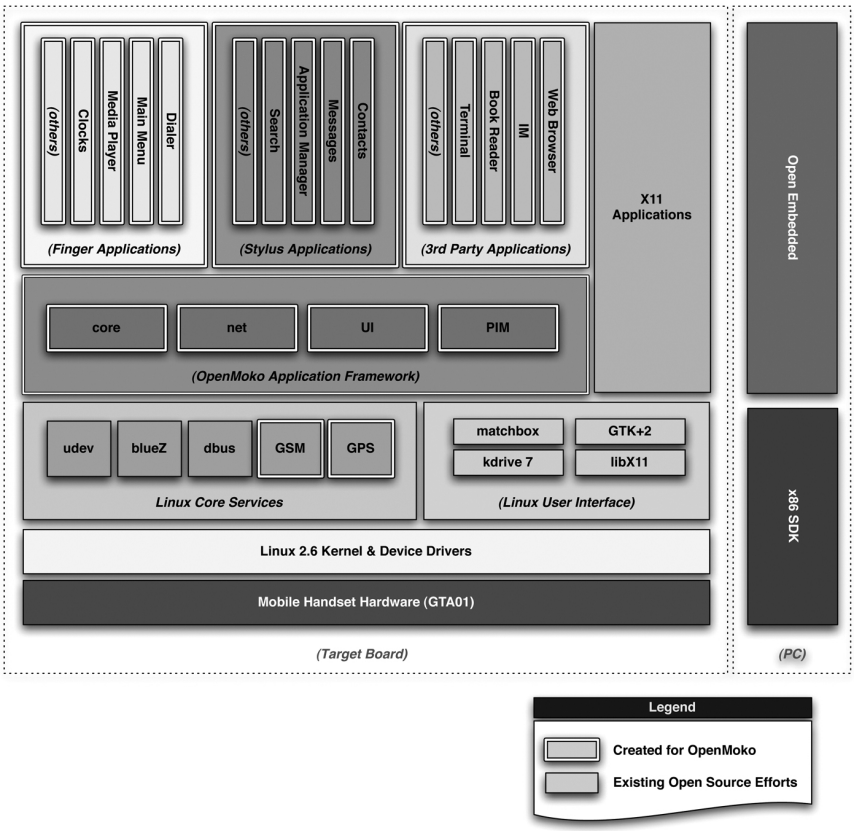

Abbildung 1: OpenMoko Softwarearchitektur

Durch die sehr frühe Veröffentlichung der Software haben sich auch schon etliche Entwickler aus der Community gefunden, die das Projekt mit Patches und Fehlerkorrekturen unterstützen.

2.3 Entwicklung

Bereits in dem gegenwärtigen Stadium der Plattform wurden schon interessante Anwendungen durch die Gemeinschaft neu entwickelt bzw. portiert – einige Beispiele dafür lassen sich auf der offiziellen Seite für Bildschirmschnappschüsse betrachten (siehe auch Abbildung 2).[7] Die schnelle Akzeptanz lässt sich auch dadurch erklären, dass eigene Programme mit Hilfe von aus dem Desktop-Umfeld bekannten Bibliotheken entwickelt werden können. Man kann eigene Anwendungen nahezu vollständig in effizienter Weise auf dem PC entwickeln und erst dann auf *OpenMoko* übertragen, wenn sie zumindest auf dem Desktop stabil laufen. Einschränkungen ergeben sich allerdings, wenn spezielle Gerätefunktionalitäten wie Telefonie, GPS, Beschleunigungssensoren etc. benutzt werden. Zur einfacheren Entwicklung ohne Gerät gibt es darüber hinaus einen speziellen Emulator, der auf *Qemu*[8] basiert. Mit diesem lassen sich dann sogar benutzerspezifische Anpassungen des Urladers, Kernel und allen weiteren Teilen der *OpenMoko* Linux-Distribution entwickeln.

Um das Ziel einer freien Softwareplattform für mobile Kommunikationsgeräte zu erreichen, setzt das OpenMoko-Team auf aus der Welt der freien Software bekannte und bewährte, transparente Entwicklungsprozesse und -modelle. Sowohl die internen als auch die externen Entwickler arbeiten dabei mit offenen Werkzeugen und unterstützen offene Standards bzw. prägen sie mit. Ein Paradebeispiel hierfür ist die Benutzung der freien Entwicklungsumgebung *OpenEmbedded*[9] (OE), mit der Software für eingebettete Systeme erstellt werden kann – damit ist es u. a. möglich, mit wenigen Kommandos zu einem kompletten OpenMoko-Systemabbild zu gelangen, das dann zur Ausführung auf das jeweilige Endgerät übertragen werden kann.

Die OpenMoko-Plattform wird derzeit im Wesentlichen von *OpenMoko Inc.* vorangetrieben – interessierte Entwickler können jedoch jederzeit Korrekturen einpflegen oder neue Funktionalitäten hinzufügen. *OpenMoko* ist ein Community-basierter Standardisierungsansatz, der sich an der Praxis – d. h. lauffähigen Anwendungen und deren Anforderungen – orientiert. Jeder kann dazu beitragen.

3 Offene Hardware für OpenMoko

Viele Projekte im Umfeld eingebetteter Systeme haben in den letzten Jahren versucht, *Linux* auf proprietärer Hardware lauffähig zu machen – leider hat sich gezeigt, dass dies ohne Unterstützung der jeweiligen Hersteller allerdings kaum zu bewerkstelligen ist. Die direkte Unterstützung des Hardwarepartners FIC bringt *OpenMoko, Inc.* in die glückliche Lage, über großes Mitspracherecht bei der Entwicklung neuer mobiler Geräte zu verfügen, auf denen *OpenMoko* eingesetzt werden kann.

7 Siehe http://scap.linuxtogo.org.
8 Siehe http://www.qemu.org.
9 Siehe http://www.openembedded.org.

Abbildung 2: Einige portierte Anwendungen auf der OpenMoko-Plattform

3.1 Neo 1973

Im Lichte der offenen OpenMoko-Plattform geht FIC auch bei der Hardware einen ähnlichen Weg und öffnet diese für interessierte Entwickler. Die Geräte sind weitreichend dokumentiert und durchaus dazu gedacht, geöffnet und untersucht zu werden – dies betont auch die Verfügbarkeit eines dedizierten Debug-Boards mit JTAG-Port, mit dem systemnahe Programmierung (z. B. Urlader- oder Kernelentwicklung) und Hardware-Experimente durchgeführt werden können. Das erste OpenMoko-Plattformgerät ist das *FIC Neo 1973* [10] (vgl. Abbildung 3) mit den folgenden Daten:

- Samsung S3C2410 CPU, getaktet mit 266MHz,
- VGA-Touchscreen mit 480x640 Pixel,

10 Martin Cooper führte 1973 das erste Telefonat über GSM.

Abbildung 3: FIC Neo1973 Mobiltelefon

- 128 MByte RAM,
- 64 MByte Flash,
- TI Calypso GSM 2.5G,
- Bluetooth 2.0,
- USB,
- MicroSD,
- AGPS.

Neue Wege wurden auch beim Entwicklungsprozess beschritten – 50 kostenlose Testmuster des *Neo 1973* wurden im Februar 2007 durch *OpenMoko Inc.* an ausgewählte Entwickler freier Software verschickt, um so erste Rückmeldungen zu bekommen. Seit Juli 2007 ist das *Neo 1973* im Direktvertrieb erhältlich, wobei es zur Zeit noch ganz ausdrücklich als Entwicklergerät beziehungsweise Kleinserie markiert ist.

3.2 Neo FreeRunner

Eine deutlich verbesserte Hardware-Version ist derzeit in Entwicklung und wird im Frühjahr 2008 in größeren Stückzahlen verkaufsbereit sein. Das Gerät mit dem Markennamen *Neo FreeRunner* basiert auf der Hardware des *Neo 1973* und verfügt über ein identisches Display und Gehäuse.

Verändert wurden jedoch einige wichtige Aspekte, die enormen Einfluss auf die Latenz und Flexibilität der Anwendungen haben:

- Samsung S3C2442B CPU, getaktet mit 400MHz,
- SMedia 2D/3D Grafikbeschleuniger,
- 128 MByte RAM, 256 MByte Flash,
- Zwei Beschleunigungssensoren,
- Zwei Leuchtdioden,
- Atheros 802.11g WiFi.

Weitere – dann deutlich stärker auf den Endkundenmarkt ausgerichtete – Hardware-Modelle sind für 2009 geplant. Neben dem schon erwähnten Navigationsgerät *Dash Express* ist mit Spannung zu erwarten, welche zukünftigen Geräte auf OpenMoko-Basis FIC als Hardwarepartner unter eigenem Namen oder auch für andere Hersteller konstruiert.

3.3 Portierungen

Durch die Offenheit der Plattform ist es nicht nur möglich, sondern seitens *OpenMoko Inc.* im Gegensatz zu anderen Herstellern sogar ausdrücklich erwünscht, die Software auch auf Geräten zum Laufen zu bringen, die nicht von FIC hergestellt wurden – letzlich kann dies nur positiv für die Verbreitung der Plattform sein. Wie Abbildung 4 zeigt, ist dies schon einigen Gruppen gelungen. Insbesondere für Geräte älterer Bauart kann dies eine interessante Alternative sein, da aufgrund des schnellen Modellwechselzyklus heutzutage oft schon die Software wenige Monate alter Geräte nicht mehr vom Hersteller aktualisiert wird.

4 OpenMoko und die „Anderen“

Im Lichte vieler Ankündigungen im Bereich quelloffener Mobiltelefonieplattformen im vergangenen Jahr 2007 stellt sich die Frage, inwieweit sich OpenMoko von ähnlichen Bestrebungen abgrenzt:

Access Linux Platform ist eine ursprünglich von *PalmSource Inc.* gestartete Initiative, um eine neue PalmOS-Plattform auf Linux-Basis zu etablieren. Das quelloffene *Hiker-Framework* enthält allerdings nur einen Bruchteil der Plattformdienste. Es gibt kaum Community-Interaktion – derzeitig ist nicht abzusehen, ob und welchen Stellenwert dieses Rahmenwerk in Zukunft haben wird.

LiMo-Foundation ist ein Konsortium aus Mobilfunkherstellern und -betreibern. *LiMo* will eine standardisierte Linux-Plattform entwicklen – dabei sollen sowohl APIs[11] definiert als auch eine Referenzimplementierung vorgenommen werden. Eine Community-Beteiligung ist nicht vorgesehen, die Mitgliedschaft ist kostenpflichtig.

11 Application Programming Interfaces, d. h. Schnittstellen für die Anwendungsprogrammierung.

Abbildung 4: OpenMoko auf Hardware anderer Hersteller

Google Open Handset Alliance ist ähnlich wie die *LiMo-Foundation* ein Konsortium, das eine standardisierte Plattform herausgeben will. Derzeit ist noch unklar, welche Bestandteile unter quelloffenen Lizenzen stehen werden und ob die Community in die Weiterentwicklung involviert werden soll.

LiPSForum ist ein Konsortium von Mobilfunkbetreibern, das standardisierte APIs herausgibt. Eine Community-Beteiligung ist derzeit nicht geplant. Die Mitgliedschaft ist kostenpflichtig.

FreeSmartPhone.org ist eine von der Community getragene Initiative, die an D-Bus–APIs sowie Referenzimplementierungen für Middleware-Dienste wie Telefonie, *Personal Information Manager* (PIM), Paketierung, Gerätekontrolle etc. arbeitet. *FreeSmartPhone.org* wird derzeit hauptsächlich von Entwicklern von *OpenMoko Inc.* sowie *Kernel Concepts* vorangetrieben. Die vom *LiPS-Forum* vorgegebenen APIs fungieren dabei als eine Grundlage. *OpenMoko* wird in künftigen Versionen die von *FreeSmartPhone.org* entwickelten Middleware–Dienste benutzen.

Maemo ist eine von *Nokia* entwickelte Plattform für Internet-Tablets (allerdings ohne GSM Telefonie). *Maemo* und *OpenMoko* basieren beide auf GNOME-Technologien, so dass sich die Arbeit an gemeinsamen Basisdiensten anbietet. Für

viele Entwickler kann es darüber hinaus attraktiv sein, zukünftig Anwendungen zu schreiben, die sowohl auf der Plattform *OpenMoko* als auch auf *Maemo* lauffähig sind.

5 Ausblick

Die OpenMoko-Plattform ist die derzeit einzige wirklich offene Plattform für Linux–basierte Mobiltelefone. FIC und *OpenMoko Inc.* konzentrieren sich derzeit darauf, Entwicklern die nötige Hard- und Softwarebasis in die Hand zu geben, um die Realisierung neuer Ideen für mobile Kommunikation und Kollaboration zu fördern. Durch die Ausrichtung auf diese Zielgruppe werden auch Initiativen aus Forschung und Lehre angesprochen, für die durchgängig programmierbare mobile Systeme mit GSM bisher nicht verfügbar waren. Schließlich weckt eine offene Mobiltelefonplattform aber auch das Interesse kleinerer Unternehmen, die etwa vertikale Märkte mit Lösungen bedienen wollen und dabei die volle Kontrolle über ihre Plattform benötigen. In der Zukunft werden FIC und *OpenMoko Inc.* weiter daran arbeiten, mit freier Software und aufregenden neuen Programmen sowie mobilen Geräten Mark Weisers Vision des allgegenwärtigen Arbeitens mit Rechnern (*Ubiquitous Computing*, siehe Weiser 1991) voranzubringen. Die Community kann dazu einiges beitragen.

Literatur

Weiser, M. (1991), 'The Computer for the 21st Century', *Scientific American* **265**, S. 94–104.

Endanwendergetriebene Open-Source-Softwareentwicklung mit Cofundos

Sören Auer

Open-Source-Software ist in vielen Bereichen überaus erfolgreich und hat eine Reihe zentraler Anwendungen und Werkzeuge hervorgebracht. Trotz dessen gibt es häufig Fälle, in denen Open-Source-Software noch nicht alle Anwenderwünsche befriedigt oder Unternehmen und Organisationen spezifische Erweiterungen zu bestehender Software benötigen. Möglichkeiten kollaborativ Anforderungen an neue Software oder die Erweiterung bestehender Software zu sammeln, zu diskutieren, zu priorisieren und letztendlich Ressourcen für deren Realisierung zu mobilisieren, sind sehr begrenzt oder nicht vorhanden. Cofundos ist ein neues Konzept für endanwendergetriebene Open-Source-Softwareentwicklung, bei dem sowohl Softwareideen, Anforderungen als auch Ressourcen in Form von Geldspenden von potenziellen Endanwendern zusammengetragen werden. Eine Plattform, die dieses Konzept implementiert, wurde unter cofundos.org bereitgestellt und der Autor berichtet über erste Erfahrungen und Schlussfolgerungen.

Schlüsselwörter: Softwareentwicklung · Softwarebörse · Online-Community · Cofundos.org · kollaboratives Arbeiten

1 Einführung

Open-Source-Software ist in vielen Bereichen überaus erfolgreich und hat eine Reihe nicht mehr wegzudenkender Anwendungen und Werkzeuge hervorgebracht. Bekannte Beispiele dafür sind der Linux-Kernel, die Bürosoftware *OpenOffice.org* oder der *Firefox*-Webbrowser.

Trotz dessen gibt es häufig Fälle, in denen noch keine passende Open-Source–Softwarelösung existiert, Open-Source-Software noch nicht alle Anwenderwünsche

befriedigt oder Unternehmen und Organisationen spezifische Erweiterungen zu bestehender Software benötigen. Die bestehenden Möglichkeiten, Anforderungen an neue Software oder an die Erweiterung bestehender Software kollaborativ zu sammeln, zu diskutieren, zu priorisieren und letztendlich Ressourcen für deren Realisierung zu mobilisieren, sind sehr begrenzt oder nicht vorhanden.

Das Cofundos-Projekt hat sich daher zum Ziel gesetzt, einen Kristallisationspunkt für neue Softwareideen und Anforderungen zu schaffen und Nutzergemeinden von Open-Source-Softwareprojekten zu befähigen, gemeinsam Ressourcen für die Realisierung der Softwareideen zu mobilisieren. Cofundos basiert dabei auf den folgenden Prinzipien: Freies Wissen und freie Software, Reputation und Gemeinschaft, Fairness und Vertrauen, große Ergebnisse mit kleinen Schritten. Der Cofundos-Prozess organisiert die Beiträge, wie Projektideen, Spendenangebote und Implementierungsvorschläge von beteiligten Softwareentwicklern und Endanwendern. Die Cofundos–Webplattform implementiert diesen Prozess und ermöglicht eine effiziente Zusammenarbeit einer Vielzahl von Beteiligten.

Wir beschreiben das Cofundos-Entwicklungsmodell anhand der zugrunde liegenden Prinzipien, des Prozesses und der Web-Plattform in Abschnitt 2, wir geben eine Übersicht über Vorteile von Cofundos für verschiedene Open-Source-Stakeholder in Abschnitt 3, wir berichten über erste Erfahrungen beim Betrieb der Webplattform Cofundos.org in Abschnitt 4, stellen verwandte Arbeiten in Abschnitt 5 vor und schließen mit Schlussfolgerungen und einem Ausblick in Abschnitt 6.

2 Das Cofundos-Entwicklungsmodell

Das Cofundos-Entwicklungsmodell basiert auf Grundprinzipien, die in einen Prozess münden. Die einzelnen Beiträge verschiedener Beteiligter zu diesem Prozess werden mittels der Internetplattform organisiert. Diese einzelnen Elemente werden in den folgenden Abschnitten detaillierter vorgestellt.

2.1 Prinzipien

Das Cofundos-Entwicklungsmodell basiert auf den folgenden Prinzipien:

Freies Wissen und freie Software. Alle Beiträge zu Cofundos werden unter die Creative–Commons-Attribution-2.0-Lizenz gestellt. Alle Projektergebnisse müssen unter eine von der Open-Source-Initiative zertifizierte Lizenz gestellt werden.

Reputation und Gemeinschaft. Innovative Ideen und exzellente Lösungen werden oft durch herausragende Einzelpersonen hervorgebracht. Es braucht allerdings eine Gemeinschaft, welche diese Ideen konkretisiert, weiterentwickelt und die eine kritische Masse an Beiträgen zu deren Realisierung mobilisiert.

Fairness und Vertrauen. Eine offene und transparente Umgebung fördert faire Kommunikation und vertrauenswürdige Beziehungen zwischen den Nutzern.

Große Ergebnisse können mit kleinen Schritten erreicht werden. Die Realisierung innovativer Softwareideen erfordert keine langjährigen Entwicklungen oder riesige Budgets. Sie kann erreicht werden, indem innovative Ideen Einzelner gemeinschaftlich mittels Anforderungen und Funktionalitäten beschrieben und durch kompetente Spezialisten implementiert werden.

2.2 Prozess

Zentrales Element des Cofundos-Entwicklungsmodells ist es, eine Vielzahl von möglicherweise sehr kleinen Beiträgen zur Weiterentwicklung einer Open-Source-Software effizient zu organisieren. Dies erfolgt im Rahmen eines Prozesses, der folgendermaßen abläuft:

Projektidee Jemand vermisst eine Open-Source-Software für eine spezielle Anwendung, ein Feature einer Open-Source-Software oder ein *plugin* für eine bestehende Software. Er schreibt ein Projekt aus, diese Software zu entwickeln.

Requirements-Engineering Andere potenzielle Nutzer der Software tragen zur spezifischeren Beschreibung bei, indem sie Anforderungen zusammentragen und diese sowie die Projektidee selbst diskutieren.

Spendengebote Potenzielle Nutzer, die Interesse am Projekt und der resultierenden Software haben, bieten an, im Falle der erfolgreichen Realisierung einen bestimmten Geldbetrag an den Entwickler der Software nach abgeschlossener und erfolgreicher Implementierung zu spenden.

Implementierungsangebote Entwickler, die die erforderlichen Fähigkeiten zur Entwicklung der Software haben, unterbreiten Angebote zu deren Implementierung in einem bestimmten Zeitrahmen für einen bestimmten Spendenbetrag.

Aufruf zu konkurrierenden Angeboten Sobald die Summe der gebotenen Spenden für die Realisierung eines Projekts den minimal von einem Spezialisten geforderten Betrag übersteigt, startet eine dreiwöchige Frist, in welcher weitere Realisierungsangebote unterbreitet werden können.

Annahme eines Angebots Nach Ablauf der Frist für alternative Angebote werden alle Spender aufgefordert, über die Annahme eines Realisierungsangebots abzustimmen. Stimmen der Spender werden bezüglich der Höhe der Spende gewichtet. Das Realisierungsangebot, welches die Mehrheit der Stimmen auf sich vereinen kann, wird für die Realisierung ausgewählt.

Abstimmung über den Projekterfolg Nachdem der Entwickler die Fertigstellung des Projekts bekannt gibt oder der im Angebot zur Realisierung festgelegte Zeitrahmen verstrichen ist, werden die Spender aufgefordert, abzustimmen, inwieweit die in der Projektbeschreibung definierten Anforderungen (auf welche der Entwickler mit seinem Angebot Bezug genommen hat) erfolgreich realisiert wurden.

Spenden an den Softwareentwickler Stimmt die Mehrheit der Spender darin überein, dass die Anforderungen erfüllt sind, werden alle Spender aufgefordert, den entsprechenden Spendenbetrag an den Entwickler zu überweisen.

Entscheidet die Mehrheit der Spender, dass die Anforderungen nur partiell erfüllt sind, wird dem Entwickler eine Verlängerung des Implementierungszeitraums gewährt, um die Implementierung zu verbessern.

Entscheidet die Mehrheit der Spender, dass die Anforderungen nicht erfüllt werden, ist das Projekt fehlgeschlagen, keine Spenden werden getätigt und das Projekt wird evtl. erneut für Gebote und Implementierungsangebote geöffnet.

2.3 Gewichtete Abstimmungen

Alle Abstimmungen in Cofundos werden bezüglich der Spendenverpflichtung des abstimmenden Nutzers gewichtet. Dies soll gewährleisten, dass Anwender mit einem besonders starken Interesse an einem bestimmten Entwicklungsprojekt auch entsprechend beim Abstimmungsprozess berücksichtigt werden.

2.4 Protokollierung

Alle Aktionen (Projektideen, Spenden, gelungene oder fehlgeschlagene Implementierungen etc.) der Cofundos-Nutzer werden protokolliert und im Nutzerprofil zusammengefasst. Dies ermöglicht Nutzern, sich über die Fähigkeiten und Verlässlichkeit anderer Nutzer zu informieren und selbst eine entsprechende Reputation aufzubauen.

2.5 Privatsphäre

Cofundos-Nutzer bestimmen selbst, inwieweit sie anonym bleiben und wieviel ihrer Identität sie offenbaren. Nutzer müssen bei der Anmeldung eine funktionierende E-Mail-Adresse angeben, diese wird jedoch zu keiner Zeit öffentlich oder anderen Cofundos-Nutzern zugänglich gemacht.

2.6 Die Plattform

Die Startseite der Cofundos-Web-Plattform, die unter http://Cofundos.org erreichbar ist, ist in Abbildung 1 dargestellt. Die Web-Plattform ermöglicht die kollaborative Bearbeitung folgender Inhalte:

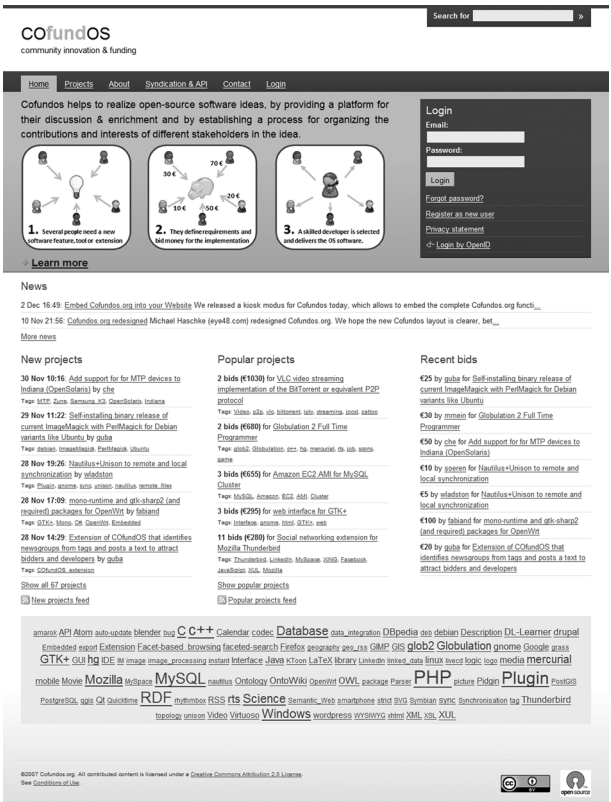

Abbildung 1: Die Cofundos.org-Startseite gibt einen Überblick über aktuelle Eintragungen.

Projektideen Die Cofundos-Plattform ermöglicht die Beschreibung von Softwareideen oder Ideen zur Erweiterung bestehender Software.

Anforderungen Projektideen können vom Initiator als auch von anderen Cofundos–Nutzern mittels Anforderungen näher beschrieben und konkretisiert werden.

Gebote Cofundos-Nutzer bieten einen bestimmten Betrag, den sie bereit sind, im Falle der erfolgreichen Realisierung eines Projekts zu spenden.

Implementierungsangebote Spezialisten, welche die erforderlichen Fähigkeiten besitzen, ein bestimmtes Projekt durchzuführen, können deren Realisierung innerhalb eines bestimmten Zeitrahmens und für einen bestimmten Spendenbetrag anbieten.

Projektideen, Anforderungen und Implementierungsangebote können von registrierten Nutzern diskutiert werden. Zur Kategorisierung der Projekte nutzt Cofundos *Tags*. Dies ermöglicht eine flexible und einfache Klassifikation von Projektideen. Die Cofundos-Web-Schnittstelle ist barrierefrei nach den Web-Accessibility-Richtlinien des *W3C* gestaltet. Eine Authentifizierung am System ist per *OpenID*[1] möglich.

Cofundos unterstützt mehrere Schnittstellen zum semantischen Datenaustausch. Projekte, Kommentare oder Gebote können als *ATOM*- oder *JSON*-Feeds bezogen werden und ein Export der kompletten Cofundos-Datenbank wird im *RDF*-Format bereitgestellt. Eine *REST*-Schnittstelle ermöglicht die Integration von Cofundos in andere Anwendungen.

Um die Nutzung von Cofundos durch Open-Source-Projekte zu stimulieren, wird ein Kiosk-Modus unterstützt, der es Projekten ermöglicht, Cofundos komplett in ihre eigene Webseite in einem angepassten Design einzubetten. Dazu können interessierte Cofundos-Nutzer eine HTML-Vorlage definieren, in welche die Cofundos-Inhalte eingebettet werden, und eines oder mehrere Tags auswählen, zu denen Projekte im Kiosk angezeigt werden sollen. Kiosks sind dann über eigene Webadressen (z. B. http://kioskname.cofundos.org) erreichbar.

3 Vorteile

Die Nutzung von Cofundos bietet für Softwareentwickler, Open-Source-Organisationen, Unternehmen und Anwender eine Reihe von Vorteilen.

3.1 Softwareentwickler

Cofundos ermöglicht es Open-Source-Softwareentwicklern, weitere Einnahmequellen zu erschließen und damit einen größeren Anteil ihrer Arbeitszeit für die Entwicklung von Open-Source-Software aufzuwenden. Perspektivisch könnte Cofundos auch helfen, insbesondere Softwareentwicklern in Schwellen- und Entwicklungsländern Verdienstquellen zu erschließen und damit deren Abwanderung in Industrieländer zu mindern. Cofundos unterstützt Softwareentwickler darüber hinaus dabei, gezielt wichtige und oft benötigte Funktionalität zu identifizieren und ihre Entwicklungsaktivitäten auf solche Projekte zu konzentrieren, die für Nutzer von besonderer Bedeutung sind.

3.2 Open-Source-Organisationen

Cofundos ist eine ideale Plattform für Organisationen, die sich der Förderung von Open-Source-Software verschrieben haben. Mittels Cofundos können gezielt wichtige förderungsbedürftige Projekte und Ideen identifiziert werden. Es könnten z. B.

1 Siehe http://openid.net.

Projekte in bestimmten Bereichen (z. B. Multimedia-Software oder Software für Behinderte) mit Entwicklern aus bestimmten Regionen oder mit bestimmten Technologien gezielt gefördert werden, indem Gebote von potenziellen Endanwendern in einem bestimmten Verhältnis aufgestockt werden.

3.3 Unternehmen

Die Vorteile von Cofundos für Unternehmen sind zweierlei: Zum einen wird Unternehmen der Einsatz von Open-Source-Software erleichtert, da die Weiterentwicklung und Anpassung der Software mittels Cofundos leichter realisiert werden kann. Für Softwareentwicklungsunternehmen bietet Cofundos darüber hinaus die Möglichkeit, sowohl als Anbieter ergänzende Einnahmequellen mittels Cofundos zu erschließen als auch Bedarfsanalysen durchzuführen und potenzielle Marktchancen für Open–Source-Softwareentwicklungsprojekte zu testen.

3.4 Anwender

Die Vorteile für Anwender von Open-Source-Software bilden den Ausgangspunkt der Cofundos-Idee. Anwendern wird durch Cofundos ermöglicht, direkter auf den Softwareentwicklungsprozess Einfluss zu nehmen. Dies soll bewirken, dass Software stärker, und bei sich neu ergebenden oder verändernden Anforderungen schneller, auf die Bedürfnisse der Anwender abgestimmt wird.

4 Erste Erfahrungen

Das Cofundos-Konzept stößt auf sehr großes Interesse. In den ersten zwei Monaten des Bestehens wurde die Cofundos.org-Webseite von mehr als 10 000 Interessenten besucht, 250 Nutzer haben sich registriert, 60 Projektideen wurden eingetragen und mehr als 6 000 Euro für deren Implementierung geboten. Eine Reihe von Medien hat über Cofundos berichtet (z. B. *heise.de*, *golem.de* und *linux.com*) und das Projekt wurde in der Blogosphäre aktiv diskutiert.

Eine Reihe von Projekten auf Cofundos wurde intensiv diskutiert und einige erste Projekte befinden sich im Realisierungsstadium. Bei diesen Projekten hat sich gezeigt, dass der finanzielle Aspekt keine allein ausschlaggebende Rolle für die Beteiligung der Softwareentwickler spielt. Angebote zur Realisierung von Projektideen werden abgegeben, weil Softwareentwickler sehen, welche Bedeutung ein bestimmtes Projekt für Anwender hat.

Trotz des großen Interesses scheint eine kritische Masse für die dauerhaft erfolgreiche Realisierung der Projektideen jedoch noch nicht erreicht. Weitere Anstrengungen müssen unternommen werden, um die Idee der endanwendergetriebenen Open–Source-Softwareentwicklung bekannter zu machen und Open-Source-Anwender noch

stärker zur Beteiligung anzuregen (siehe dazu auch den Ausblick zur Cofundos-Weiterentwicklung in Abschnitt 6).

5 Verwandte Arbeiten

Es gibt bereits eine Reihe von Diensten im Web, die es Besuchern ermöglichen, Ideen und Geld für das Erreichen eines bestimmten Ziels zu sammeln. Beispiele hierfür sind *Pledgebank*, *Fundable*, *Change.com* oder einige *Facebook*-Applikationen. Dabei geht es vor allem um karitative, soziale oder kulturelle Projekte. Umfangreiche Abstimmungsprozesse oder spezifische Unterstützung für Softwareentwicklungsprojekte sind bei diesen Diensten nicht vorhanden.

Eine Plattform, die etwa zeitgleich mit Cofundos startete und ähnlich wie Cofundos auf Softwareentwicklung abzielt, ist *Micropledge*[2]. Anders als Cofundos ist *Micropledge* nicht auf Open-Source-Software ausgerichtet und unterstützt auch kommerzielle Softwareentwicklungen. *Micropledge* ist jedoch weitaus weniger offen (keine transparenten Abstimmungsprozesse, keine freie Lizenzierung der Beiträge) und etwas komplizierter, da Projekte in Teilprojekte (mit Teilzahlungen) unterteilt werden können. Darüber hinaus gibt es natürlich eine Reihe von Vermittlungsplattformen, die sich auf (Open-Source-)Softwareentwicklung und Dienstleistungen spezialisiert haben (z. B. *OpenSourceExperts.com* oder der *SourceForge Marketplace*), denen jedoch der Aspekt des *pooling* von Ideen und Ressourcen fehlt.

Unter dem Begriff Bürgerhaushalt (*participatory budgeting*) werden eine Reihe von Strategien und Initiativen zusammengefasst, deren Ziel es ist, Bürger direkter in Haushaltsentscheidungen einzubeziehen. Die Verwaltung einer Stadt, einer Gemeinde oder einer anderen Verwaltungseinheit bemüht sich dabei z. B. um mehr Haushaltstransparenz und lässt die Betroffenen zumindest über einen Teil vom Investitionshaushalt mitbestimmen und entscheiden. In Porto Alegre (Brasilien) wurde der Beteiligungshaushalt 1989 erstmals mit breiter Bürgerbeteiligung durchgeführt (Herzberg 2006). Im Rahmen des Spendenparlaments in Hamburg[3] wird abgestimmt, welche karitativen Projekte durch Spenden unterstützt werden sollen. Ein weiteres Beispiel ist *The People's 50 Million Pound Lottery Giveaway*[4], eine Online-Abstimmung über die Vergabe von 50 Millionen britischer Pfund aus Lotterieerlösen in Großbritannien.

6 Schlussfolgerungen und Ausblick

Die in den ersten Monaten gewonnenen Erfahrungen zeigen, dass das Cofundos–Entwicklungsmodell großes Interesse und Zuspruch findet, jedoch noch einige Zeit,

2 Siehe http://micropledge.com.
3 Siehe http://www.spendenparlament.de.
4 Siehe http://www.thepeoples50million.org.uk.

einiges Experimentieren und Erweiterungen der Plattform notwendig sind, damit dieses Entwicklungsmodell dauerhaft erfolgreich etabliert werden kann. Eine Strategie, das Entwicklungsmodell noch stärker publik zu machen, ist, Stiftungen und institutionelle Förderer von Open-Source-Software zu überzeugen, Gebote zur Realisierung von Projekten auf Cofundos finanziell aufzustocken. Ziel ist es, dadurch ein stärkeres Interesse von Projekten zu wecken, Cofundos gezielt für das Requirements-Engineering zur Weiterentwicklung ihrer Software zu nutzen.

Eine technische Weiterentwicklung, die ebenfalls auf eine weitere Verbreitung der Cofundos-Idee abzielt, ist die direkte Integration von Cofundos in Bugtracking-Feature-Request-Systeme (z. B. *Mantis* oder *Bugzilla*). Anwender sollten dabei direkt, wenn sie eine neue Anforderung oder einen Fehlerreport in diesen Systemen anlegen, die Möglichkeit haben, einen bestimmten Betrag für dessen Realisierung mittels Cofundos zu spenden. Eine weitere Maßnahme, Anwender direkt zu ermutigen, Projektideen in Cofundos einzutragen, ist die Integration von Verweisen auf Cofundos direkt aus den Softwarewerkzeugen, z. B. über einen Menüpunkt „Neue Funktion anfordern" im Hilfemenü.

Im vom Bundesministerium für Bildung und Forschung im Rahmen der Forschungsoffensive *Software Engineering 2006* geförderten Projekt *SoftWiki*[5] wird im Moment untersucht, wie agiles Requirements-Engineering für Softwareprojekte mit einer großen Anzahl verteilter Stakeholder effizient realisiert werden kann (vgl. Auer et al. 2006). Geplant ist, Cofundos im Rahmen dieses Projekts weiter auszubauen und Ergebnisse des Projekts in die Cofundos-Weiterentwicklung einfließen zu lassen.

Cofundos sieht sich als ersten Schritt in Richtung von *Participatory R&D*, einer Entwicklung zu mehr Endanwenderbeteiligung und inkrementeller Forschung und Entwicklung. Insbesondere weite Teile angewandter Forschung könnten von Cofundos-ähnlichen Anforderungs-, Priorisierungs- und Evaluationsmechanismen maßgeblich profitieren.

7 Danksagung

Ich möchte dem Aktiven Leo Wandersleb aus der Cofundos-Gemeinschaft, den Diskussionsteilnehmern der GWTF-Mailingliste wie auch meinen Kollegen Thomas Riechert, Sebastian Dietzold und Jens Lehmann der Forschungsgruppe *Agile Knowledge Engineering and Semantic Web*[6] (AKSW) für ihre Unterstützung und anregende Diskussionen zum Thema Open-Source-Software, *participatory budgeting* und Software–Engineering danken. Das Projekt Cofundos und diese Publikation wurde unterstützt durch das BMBF (SE2006 #01ISF02B).

5 Siehe http://www.softwiki.de.

6 Siehe http://aksw.org.

Literatur

Auer, S., Riechert, T. und Fähnrich, K.-P. (2006), SoftWiki - Agiles Requirements-Engineering für Softwareprojekte mit einer großen Anzahl verteilter Stakeholder, *in* K. Meißner und M. Engelien (Hrsg.), 'Proc. Workshop Gemeinschaften in Neuen Medien (GeNeMe)', TUDPress Verlag der Wissenschaften, Dresden.

Herzberg, C. (2006), *Der Bürgerhaushalt von Porto Alegre. Wie partizipative Demokratie zu administrativen Verbesserungen führen kann.*, Lit-Verlag, Münster.

Kapitel 2

Von der Innovation zum Geschäftsmodell

„Nur die Kosten zu senken, ist keine Strategie."

– Henning Kagermann

Open Source als Werkzeug der Wirtschaft

MATTHIAS CHOULES UND ROMAN RAUCH

„Wir müssen uns Sisyphos als einen glücklichen Menschen vorstellen."
(Camus 2000, S. 160)

Der Vergleich mit dem *Mythos des Sisyphos* liegt nahe, versucht man Open Source in der Wirtschaft zu platzieren: Kann es Sinn ergeben, Zeit und damit auch Geld zu investieren, für etwas, das auf den ersten Blick höchstens moralischen Gewinn bringen wird und damit wirtschaftlich keinen *Sinn* ergibt?

Der Mensch treibt seit jeher Handel und wirtschaftet gewinnorientiert, stets bemüht, durch überzeugende Qualität oder günstige Produkte der Konkurrenz einen Schritt voraus zu sein. Um einen solchen Vorsprung zu halten oder aber einen entsprechenden Rückstand wieder aufzuholen, bedarf es gewisser, nach Möglichkeit innovativer Konzepte und Strategien. Diese sind aber, sollte man meinen, nur so lange wirkungsvoll, wie sie exklusiv genutzt werden. So lag es nahe, dass „Unternehmen" schon vor sehr langer Zeit viel Wert darauf legten, entscheidende Verfahren und wertvolles Wissen durch Geheimhaltung für sich zu bewahren. Spätestens seit im Jahr 1474 in Venedig die ersten Patentgesetze im heutigen Sinne erlassen wurden (vgl. Schippel 2001), ging man immer mehr dazu über, Innovationen auch durch Patente zu schützen.

Mit der wachsenden Popularität von Open Source rückt nun auch eine weitere, naheliegende Strategie in den Fokus vieler Unternehmen: eine für die Öffentlichkeit frei zugängliche Entwicklung neuer Produkte, die die Partizipation Dritter nicht nur ermöglicht, sondern sogar erwünscht. Mittlerweile etablierte Open-Source-Lizenzmodelle ermöglichen es, die Verwertungsrechte dieser Produkte so weit einzuschränken, dass Konkurrenten aus ihnen keinen kommerziellen Nutzen ziehen können bzw. dürfen, obwohl alle notwendigen Ressourcen frei zugänglich sind.

Es wird kostbares Wissen preisgegeben, einem jeden unter gewissen Konditionen zugänglich gemacht und zeitgleich verhindert, dass die Konkurrenz dieses Wissen für den eigenen Vorteil nutzen kann. Schachspieler verwenden für eine solche Strategie

den eleganten Begriff „Gambit" oder auch „Bauernopfer". Ist der absehbare Entwicklungsvorsprung groß genug, zahlen sich Investitionen in solche Open-Source-Projekte schnell aus und geben der Sisyphos-Arbeit wieder einen Sinn.

Dass dieser Entwicklungsvorsprung sehr weit ausgebaut werden kann, zeigt Matthias Bärwolff in seinem Artikel „Monopolelemente bei freier Software". Hierbei geht er auf Möglichkeiten von Freie-Software-Projekten ein, durch gezielte Restriktionen in der Entwicklung Marktmacht auszuüben und auf diese Weise eine gewisse Monopolstellung zu behaupten.

Über die Marktposition hinaus gibt es noch weitere Beweggründe, auf ein Open–Source-Geschäftsmodell zu setzen – einige werden von Joel West in seinem Artikel „Unternehmen zwischen Offenheit und Profitstreben" erläutert. Besonderes Augenmerk wird in diesem Beitrag auf die Tatsache gelegt, dass der Ausdruck *open* mittlerweile in vielen Fällen für Marketingzwecke missbraucht wird und nicht die tatsächliche *Offenheit* eines Unternehmens oder seiner Produkte widerspiegeln muss.

Eben diese *Offenheit* von Unternehmen kann auch gänzlich anders geartet sein. So untersuchen Klaus-Peter Wiedmann, Lars Pankalla und Sascha Langner in ihrer gemeinsamen Arbeit für dieses Kapitel den Trend hin zum Open-Source-Marketing und was Konsumenten dazu bewegt, sich freiwillig in die Vermarktungsprozesse von Unternehmen einzubringen.

Die Motivation hinter Open-Source-Bestrebungen von Unternehmen erörtert Thorsten Busch in seinem Artikel „Open Source und Nachhaltigkeit" aus wirtschaftsethischer Sicht und geht dabei im Besonderen auf den *Digital Divide* ein. Müssten Unternehmen nicht versuchen, durch Öffnen ihrer (Software-)Produkte für mehr Gerechtigkeit zu sorgen?

Und wenn sich Unternehmen nun partout dagegen wehren? Eine solche Situation wurde 2007 vom Europäischen Gericht erster Instanz entschieden. Auf dieses Urteil und den Verlauf des Verfahrens geht Leonie Bock in ihrem Beitrag „Der Fall Microsoft – Offengelegte Schnittstellen und offengebliebene Fragen" ein. Das Öffnen der Schnittstellen zum *Microsoft* Betriebssystem *Windows* wird auf jeden Fall auch den zahlreichen Open-Source-Projekten die Arbeit erleichtern und letztlich auch vielen Anwendern zugutekommen.

Um mit einem Wort von Camus (2000) zu schließen: „Der Kampf gegen Gipfel vermag ein Menschenherz auszufüllen." – und bringt zuweilen überraschende Ergebnisse.

Literatur

Camus, A. (2000), *Der Mythos des Sisyphos*, Rowohlt Taschenbuch Verlag, Reinbek.

Schippel, H. (2001), Die Anfänge des Erfinderschutzes in Venedig, *in* U. Lindgren (Hrsg.), 'Europäische Technik im Mittelalter, 800 bis 1400, Tradition und Innovation', 4. Aufl., Gebr. Mann Verlag, Berlin, S. 539–550.

Monopolelemente bei freier Software

MATTHIAS BÄRWOLFF*

(CC-Lizenz siehe Seite 281)

Die Quelltexte freier Software sind zwar unter der GPL-Lizenz uneingeschränkt verfügbar, andere Kontexte sind jedoch weit mehr ausschließbar gegenüber Dritten. Die Kontrolle über ein Freie-Software-Projekt und dessen spezifische Ressourcen führt dazu, dass auch freie Software Marktmacht ausüben kann. Dies ist nicht moralisch verwerflich, sondern im Gegenteil unabdingbare Voraussetzung für die Möglichkeit, Profite auf Seiten der Produzenten zu erwirtschaften und unabhängig von Subventionen zu sein. Kategorien von Software, bei der keine Möglichkeiten bestehen, Profite mit freier Software zu erwirtschaften, werden demnach traditionell überwiegend von proprietären Anbietern dominiert, da diesen über Lizenzgebühren ein Teil des erwirtschafteten Mehrwerts zufällt.

Schlüsselwörter: Freie Software · Monopol · Monopolistischer Wettbewerb

1 Einleitung

Freie Software wird häufig als der Antipol schlechthin zu proprietärer Software verstanden. So bemerkt etwa Moglen (2007):

> „Die Qualität von Software hat von 1980 bis 1990 rapide abgenommen, [. . .] [denn] Monopole produzieren bekanntermaßen minderwertige Güter zu hohen Kosten und verhindern Innovationen. [. . .] [Aber] eine kleine und unorganisierte Community von Leuten, die Software zur *gemeinsamen* Nutzung entwickelten, hat die Begrifflichkeiten der Debatte neu definiert, vortreffliche Güter zu Kosten von Null produziert und mit der nunmehr unaufhaltsamen Auflösung des Monopols begonnen."

* Ich danke Wolfram Riedel für seine scharfsinnigen und außerordentlich hilfreichen Anmerkungen zu einer früheren Version dieses Artikels.

Freie Software und „das Monopol" sind demnach unüberbrückbare Gegensätze – zwei Pole eines Kontinuums, mithin ohne jedwede Schnittmenge. Die Prämissen dieser Aussagen sind in etwa die folgenden: Freie Software unterliegt per Definition der *GNU General Public License* (GPL) oder einer anderen Open-Source-Lizenz, die die fortwährende Verfügbarkeit des Quelltextes von Software sicherstellt.[1] Mithin ist also jeder Versuch, aus dem Verkauf von solcherart lizenzierter Software Kapital zu schlagen, zum Scheitern verurteilt. Schließlich erlaubt es die Lizenz jedermann, die Software weiterzugeben, wenn auch zu denselben Lizenzbedingungen. Der Marktpreis wäre also ob der praktisch unbegrenzten Verfügbarkeit der Software gleich den marginalen Kosten des Vertriebs. Was die Nutzer der Software angeht, so könnten diese die Software nach Belieben verändern oder Dritte die Software verändern lassen. „Lock–ins", also das Ausgeliefertsein an einen bestimmten Anbieter oder eine bestimmte Technologie, sind daher ebenso abwegig wie das Erzielen von Monopolerlösen aus dem Verkauf von Software.

Aber schon Lessig (2002) hat bemerkt, dass die Produktion von freier Software insoweit vergleichbar ist mit anderen Produktionsprozessen, als dass bei ihr private Ressourcen mit öffentlich verfügbaren Allmenderessourcen zur Schaffung von ökonomischem Mehrwert vermischt werden. Dabei landen zumindest Teile des Mehrwerts bei den Produzenten. Wir möchten in diesem Artikel noch einen Schritt weiter gehen und fragen: Sind freie Software und Monopolmacht tatsächlich unvereinbar? Sind Freie-Software-Projekte vielleicht letztlich gar nicht so verschieden von proprietärer Software, als dass sie ebenso Marktmacht entwickeln und ausüben können?

Unsere Intuition dabei ist ebenso einfach wie die oben beschriebene und hier in Frage gestellte. Eine Open-Source-Lizenz führt zwar zur unbeschränkten Verfügbarkeit des *Codes*, nicht aber der von spezifischen *Kontexten*. Hierzu zählen schon allein der Name einer Software, dessen markenrechtlicher Schutz ganz unabhängig von der Nutzungslizenz ist, aber auch die Beherrschung von komplementären Fertigkeiten oder die Existenz bestimmter komplementärer Artefakte auf Seiten von Produzenten und Anwendern. Daraus folgt, dass es sowohl einen Ort der Kontrolle über ein jedes Freie-Software-Projekt gibt als auch eine daraus resultierende Marktmacht sowie das Potenzial, mittels Komplementäreffekten Monopolrenten zu erwirtschaften.

2 Freie Software im Kontext von monopolistischem Wettbewerb

In diesem Abschnitt schweifen wir kurz ab und betrachten die wechselseitige Abhängigkeit von Wettbewerb und Monopolen, um das Konzept des monopolistischen Wettbewerbs später auf freie Software anwenden zu können. Die häufig gezeichnete Dichotomie zwischen den Begriffen Wettbewerb und Monopol ist nämlich – genauso

1 Wir beschränken uns im Rahmen dieses Artikels auf die Betrachtung GPL-lizenzierter Software und verzichten auf eine Ausdehnung der Betrachtung auf BSD-artige Lizenzen oder andere Lizenzen, die von der *Open Source Initiative* anerkannt sind (http://www.opensource.org/licenses).

wie deren populäre ethische Bewertung – nicht nur fragwürdig, sondern häufig auch irreführend.

Monopole sind in einem ganz elementaren Sinne entscheidend für das Funktionieren einer dezentralen Wirtschaft. Wettbewerb und Monopole bedingen einander, wie Chamberlin (1950) treffend ausführt:

> „Ein wesentlicher Teil freien Unternehmertums ist das Bemühen eines jeden Geschäftsmannes, [nicht homogene Güter in vollkommenem Wettbewerb zu produzieren, sondern] sein eigenes Monopol aufzubauen, es möglichst weit auszudehnen und es zu verteidigen gegen die Bemühungen anderer, ihre Monopole auszubauen. Es gibt keine Tendenz, derzufolge diese Monopole durch den Wettbewerb zwischen Unternehmen verschwinden, ganz im Gegenteil: Sie sind genauso ein Teil des Gesamtbilds wie der Wettbewerb, der sie beschränkt.
>
> Die Tatsache, dass Produkte differenziert sind, bringt die Probleme der Vielfalt und Auswahl zum Vorschein und macht deutlich, dass *vollkommener Wettbewerb in keinster Weise als ‚ideal' anzusehen ist für volkswirtschaftliche Wohlfahrtsbetrachtungen*. In vielen Fällen wäre es praktisch unmöglich, solchen Wettbewerb herzustellen, selbst wenn er denn wünschenswert wäre. Geschäfte, zum Beispiel, könnten sich nicht alle an einem Ort befinden, genauso wie Unterschiede zwischen Schauspielern, Sängern, Freiberuflern und Geschäftsleuten sich nicht vermeiden lassen. Und selbst dort wo es möglich wäre, wäre es nicht erstrebenswert, Produkte übermäßig zu standardisieren. Verschiedene Geschmäcker, Vorlieben, Einkommen, Wohnorte und Nutzungen von Gütern bezeugen schließlich die Notwendigkeit von Produktvielfalt. Das ‚Ideal des vollkommenen Wettbewerbs' muss also ersetzt werden durch eines, das beides beinhaltet, Monopol und Wettbewerb, und dies unbeschadet der Frage, wie viel und welche Art von Monopol, mit welcher Art von sozialer Kontrolle.
>
> Weiterhin ist es unerlässlich, die ‚ideale' Abstimmung der Kosten für Herstellung, Werbung und Vertrieb, wie die konventionelle Analyse von Preis und Produktionsmenge auch, explizit als Teil des Wohlfahrtsoptimums zu begreifen. Diese zusätzlichen Elemente sind fraglos weitaus eher veränderlich als die Preise selbst, und doch ist es die Standardannahme der ökonomischen Wohlfahrtsmaximierung, dass erstens Produkte gegeben sind und es zweitens keine Vertriebs- und Werbekosten gibt, [was natürlich falsch ist]." (S. 213 ff., Fußnoten weggelassen, Betonung im Original, Übersetzung des Autors)

Monopole sind also nicht etwas, das es „natürlicher Weise" in einer Wirtschaft nicht gibt, sondern im Gegenteil gerade eine Folge von Wettbewerb unter verschiedenen Anbietern.

Wettbewerb wird nur in den Lehrbüchern vereinfacht gleichgesetzt mit festgelegten Gütern, deren Preise sich dann im Wettbewerb den marginalen Kosten nähern. In der Realität gibt es vielmehr eine Reihe weiterer veränderlicher Variablen, über die sich Wettbewerb definiert, unter anderem eben die Produkte selbst.[2] Weiterhin werden in idealisierten Betrachtungen Kosten für Marketing, Vertrieb und Werbung entweder als unnütz und schädlich abgetan oder gleich gänzlich vernachlässigt. Damit unterschlägt man jedoch eine in der Realität immens wichtige Angelegenheit. Über strategische Entscheidungen und Aufwendungen in diesen Bereichen entscheiden sich häufig Erfolg oder Misserfolg eines Unternehmens, nicht über die Preisgestaltung entlang hypothetischer Nachfragekurven für festgesetzte Produkte.

Dass dies im Bereich der freien Software ökonomisch betrachtet anders sein soll, ist zwar eine intuitiv verlockende Vorstellung, empirisch jedoch völlig unhaltbar.

Heute ist weitgehend unstrittig, dass freie Software, genau wie jedes andere Gut auch, entweder durch Monopolrenten oder durch Subventionen finanziert wird. Die Möglichkeit für Produzenten von freier Software, aus dem untrennbaren Kontext einer Software Profite zu erzielen, also aus Komplementärgütern oder -dienstleistungen, wie spezieller Anpassung der Software oder Unterstützung bei deren Einrichtung und Betrieb, wurde in der Literatur sehr ausführlich beschrieben (siehe etwa Grand et al. 2004; Bärwolff 2006) und braucht hier nicht weiter erörtert zu werden.[3]

Freie Software, deren Entwicklung und Vertrieb nicht durch solche Monopolrenten finanziert wird, ist häufig finanziert durch Steuersubventionen – also öffentliche Aufwendungen im universitären oder anderweitig steuerfinanzierten Forschungsbereich – oder durch Firmen, deren kommerzielle Software im Markt gescheitert ist und die daraufhin ihre Software als freie Software veröffentlicht haben. Eine wichtige Rolle spielen zudem Entwickler, deren Opportunitätskosten hinreichend gering sind (Lancashire 2001) und die durch ihre Mitarbeit an freier Software in einer Art „zeitlich umgekehrter Quersubventionierung" soziales Kapital akkumulieren und potenziellen Arbeitgebern ihre Fähigkeiten signalisieren (Lerner und Tirole 2002).[4]

Im folgenden Abschnitt möchten wir diesen empirischen Befunden folgen und zeigen, dass freie Software in vielerlei Hinsicht sehr ähnlich zu proprietärer Software ist. Schon die Produktion freier Software ist häufig weit weniger frei, als der Name es erscheinen lässt, was dazu führt, dass auch in den entsprechenden Produktmärkten Monopolmacht ausgeübt werden kann.

2 Bei Kosten von Null (wie im Falle von GPL-lizenzierter Software) wären ja sonst *alle* Variablen per Definition gegeben – eine völlig absurde Vorstellung.

3 Wir möchten nur kurz anmerken, dass es in diesem Bereich interessante Entwicklungen gibt, etwa die viel diskutierte TiVo-Festplatten-Set-Top-Box, bei der die GPL-lizenzierte Software des Geräts derart mit der Hardware verknüpft ist, dass diese keine modifizierten Versionen der Software akzeptiert. Die Freiheiten der GPL sind hier, zumindest im Bezug auf die TiVo-Hardware, nur theoretisch gegeben, praktisch aber irrelevant, da Änderungen der Software zwar möglich sind, nicht aber auf der Zielplattform lauffähig.

4 Siehe zur Bedeutung von geringen Opportunitätskosten für informelle Produktionsprozesse auch Pinker (2006).

3 Freie-Software-Projekte als Firmen

Um unserem Argument einen tragfähigen Ausgangspunkt zu geben, ist es zunächst sinnvoll, festzuhalten, dass es auch bei freier Software einen institutionellen und organisatorischen Kontext gibt, in dem Kontrolle über die Entwicklung und den Vertrieb einer Software ausgeübt werden kann und oft auch wird. Insofern unterscheidet sich ein Freie-Software-Projekt nicht großartig von einem kommerziell motivierten proprietären Softwarehersteller. Beides sind Firmen im Sinne von Coase (1937), als dass es einen mehr oder minder definierten Raum gibt, in dem die Allokation von Ressourcen nicht über Preise, sondern durch Befehlsketten in Hierarchien organisiert wird. In kommerziellen Unternehmen sind es die Aktionäre, mittelbar vertreten durch die Führungsebene, die die angestellten Mitarbeiter anleiten und kontrollieren. Bei freier Software ist es zumeist der ursprüngliche Entwickler, der zum einen die Namensrechte an der Software hält und zum anderen die Kontrolle und Führung über das Projekt ausübt.

Wenn die freie Software einer kommerziellen Firma „gehört“,[5] dann gilt prinzipiell selbes wie im Kontext einer jeden anderen Firma: Bezahlte Angestellte entwickeln die Software auf Weisung ihres Arbeitgebers. Nur selten gibt es dann überhaupt externe Parteien, die Rechte irgendwelcher Art an dem resultierenden Produkt haben. Externe Beiträge zur Software werden entweder nicht berücksichtigt oder nur aufgenommen, wenn die Entwickler ihre Rechte daran der Firma überschreiben.[6] Und auch, wenn die freie Software von einer verteilten Community außerhalb eines formalen Firmenkontexts entwickelt wird, gibt es praktisch immer eine Hierarchie, in der die oberen Ebenen Kontrolle über die unteren ausüben, zumindest aber darüber, welche Arbeiten ihren Weg in die freie Software des Projekts finden (Priddat 2006). Es ist mithin nicht unüblich, dass Beiträge von Entwicklern abgelehnt werden, und dies nicht etwa, weil sie anderen konkurrierenden Beiträgen qualitativ nachstehen, sondern weil es gerade bei großen Projekten überhaupt nicht mehr ohne Weiteres möglich ist, die Qualität von Beiträgen objektiv und nach rein technischen Kriterien zu beurteilen. Status wird dann häufig wichtiger als Leistung, ganz entgegen der gängigen Behauptung, in Freie-Software-Projekten würden Beiträge *alleine* nach objektiven technischen Qualitätskriterien beurteilt.[7] In Freie-Software-Projekten entstehen also die selben Ineffizienzen wie in jeder anderen Firma auch, sie entkommen nicht der

5 Siehe etwa das MySQL-Datenbankmanagementsystem der schwedischen Firma *MySQL AB* oder den Java-Application-Server *JBoss* der mittlerweile zu *Red Hat, Inc.* gehört.

6 Für *MySQL AB* ist es zum Beispiel nur dann möglich, externe Beiträge mit in die MySQL-Software aufzunehmen, wenn die externen Entwickler ihre Rechte daran an MySQL abtreten. Das MySQL–Geschäftsmodell, die Software sowohl unter der GPL als auch unter proprietären Lizenzbedingungen zu vermarkten, bedingt es, dass alle Rechte an der Software bei *MySQL AB* liegen.

7 O'Mahoney und Ferraro (2004) betrachten das Debian-Projekt und dessen Probleme, mit einer wachsenden Mitgliederzahl umzugehen. An einem Punkt wurde das Projekt gar völlig geschlossen für neue Mitglieder, als klar wurde, dass die Organisation des Projekts nicht einmal mehr operativ, geschweige denn effizient mit der wachsenden Mitarbeiterzahl umgehen konnte.

Logik der Coaseschen Transaktionskosten, nur weil sie sich anderen Idealen als der Profitmaximierung verpflichtet fühlen.[8]

Ein häufig geführtes Argument besagt nun, ein jeder könne ob der frei vorhandenen Quelltexte einem Freie-Software-Projekt Konkurrenz machen, etwa falls dieses sich zu sehr kommerziellen Interessen beugt oder in anderer Weise „unfreier" wird, aber auch falls jemand das Projekt einfach nur in eine Richtung weiterentwickeln möchte, die das Projekt ablehnt. Durch diese prinzipiell vorhandene Möglichkeit des *forking*[9] löst sich jedoch nicht das eben beschriebene Problem. Zum einen sind Entwickler für freie Software nicht unbegrenzt verfügbar,[10] und zum anderen gelten auch für das neue Projekt dieselben strukturellen Einschränkungen der Freiheit wie für das alte. Auch hier wird jemand die Kontrolle über das Projekt ausüben, auch hier wird es eine Entscheidungshierarchie geben, und auch hier werden entsprechende Ineffizienzen auftreten.

Dazu kommt, dass die geistigen Eigentumsrechte an der Software, zumindest bis zum Punkt der Abspaltung des neuen Projekts, bei den ursprünglichen Entwicklern verbleiben, auch wenn die Software unter einer Open-Source-Lizenz jedem frei zur Verfügung steht. Das bedeutet mithin, dass derjenige, der das neue Projekt ins Leben ruft, nur eben jene Rechte an der freien Software erwirbt, die ihm qua Open-Source–Lizenz zustehen.[11] Weiterhin wird die neue Software sowohl aus praktischen Gründen als auch aus markenrechtlichen Gründen unter einem neuen Namen firmieren müssen, dem ursprünglichen Projekt also nicht unter demselben Namen Konkurrenz machen können. So besteht also die prinzipielle Möglichkeit, einem Freie-Software-Projekt Konkurrenz mit dessen eigenen Quelltexten zu machen, es besteht aber auch ein sehr wirkungsvoller und begrenzter Ort der Kontrolle über ein Projekt, der sich nur schwer untergraben lässt. Wenn es nun also bei freier Software de facto einen „Eigentümer" gibt, der oft sogar eine juristische Person ist, dann gibt es damit auch jemanden, der Monopolmacht ausüben kann. Die Frage ist nun, ob und in welcher Weise solche Monopolmacht ausgeübt wird.[12]

8 Mitunter wird argumentiert, die Verteilung von Aufgaben in Freie-Software-Projekten passiere mehr oder minder selbstorganisiert, indem offene Aufgaben als solche gekennzeichnet werden und dann von Entwicklern mit freien Kapazitäten übernommen werden, ohne dass es hierfür einer zentralen Koordinierung bedürfe (Heylighen 2007). Dies ändert jedoch nichts an den institutionalisierten hierarchischen Kontrollprozessen in Freie-Software-Projekten, über die sich entscheidet, welche Beiträge in das Projekt aufgenommen werden.

9 Mit *forking* bezeichnet man das Abspalten eines neuen Software-Projekts aus einem gegebenen Projekt.

10 Anders ausgedrückt, die Ressourcen eines Projekts und die Fähigkeiten der Mitarbeiter sind ob ihrer Spezifizität nicht beliebig kopierbar (Wernerfeld 1984).

11 Unter den Lizenzbedingungen der GPL etwa kann man, da die neue Software ein abgeleitetes Werk der alten bleibt, nur die Erweiterungen, nicht aber das gesamte Werk unter einer anderen Lizenz weitervertreiben. Dies ist beispielsweise die Basis des Geschäftsmodells der oben schon erwähnten Firma *MySQL AB*. Falls jemand die MySQL-Software erweitern und unter einer anderen Lizenz als der GPL weitergeben möchte, so kann er hierfür die Software statt unter der GPL auch unter einer proprietären Lizenz von *MySQL AB* erwerben, da diese ja sämtliche Rechte an der Software hält.

12 Die Tatsache, dass freie Software praktisch immer zu Kosten von Null abgegeben wird, tut dieser

4 Zwei kurze Fallbeispiele für Monopolelemente bei freier Software

Sobald ein Freie-Software-Projekt eine gewisse Marktrelevanz entwickelt, beginnt es den obigen Ausführungen zufolge, über potenzielle Marktmacht zu verfügen.[13] Zwei Fallbeispiele mögen dies veranschaulichen:

Firefox vs. Iceweasel Der Webbrowser *Firefox* der *Mozilla Foundation* ist fast schon ein Paradebeispiel für die Monopolmacht von Produzenten freier Software. Trotz Kosten von Null für die gegebene freie Software werden hier Profite erzielt und Konkurrenten diskriminiert: Profite werden erzielt durch die Partnerschaft mit *Google*, die jährlich etwa 50 Millionen US-Dollar an Einnahmen bringt.[14] Und Konkurrenten, die auf Basis der Quelltexte von *Firefox* unter der GPL–Lizenz das Produkt ändern und weitervertreiben möchten, sind an strenge markenrechtliche Richtlinien der *Mozilla Foundation* gebunden, denen zufolge niemand die Quellen des Browsers ändern und dann unter dem Namen *Firefox* weitergeben darf. Das Debian-Projekt etwa, eine der ältesten und bedeutendsten Linux-Distributionen weltweit, ist dadurch in Konflikt mit der *Mozilla Foundation* geraten. Als man im Jahre 2006 kleine Verbesserungen und Anpassungen an *Firefox* vornehmen wollte, deren Aufnahme in das Hauptprojekt von den Mozilla-Entwicklern abgelehnt wurde, verbot die *Mozilla Foundation* es dem Debian-Projekt, die resultierende Software weiterhin unter dem Namen *Firefox* in ihre Linux-Distribution aufzunehmen. Seither nennt sich diese Software bei *Debian Iceweasel*, eine für Anwender zunächst gewöhnungsbedürftige Änderung. Nun sind *Iceweasel* und *Firefox* zwar noch zum ganz überwiegenden Teil dieselbe Software, es sind aber eben doch unterschiedliche Produkte mit unterschiedlichen Namen, unterschiedlichen Vertriebswegen und mithin unterschiedlichen Anwendergruppen.

Dieses Beispiel entspricht ziemlich genau dem Modell des monopolistischen Wettbewerbs von Chamberlin (1950). Die *Mozilla Foundation* möchte verhindern, dass andere Anbieter ihr Monopol auf den Browser *Firefox* untergraben,

---

Frage keinen Abbruch. Schon Chamberlin (1950) bemerkte sehr treffend, dass sich Monopole nicht unbedingt alleine über Preise, schon gar nicht über Profite definieren:

> „[P]rofite sind lediglich ein Element in [Monopol-]Situationen; Preise, Diskriminierungspraktiken, Dienstleistungen in allen ihren Aspekten, Investitionen usw. können stark beeinflusst sein durch Monopolelemente, obgleich keine exzessiven Profite erzielt werden." (S. 195 f., Übersetzung des Autors)

13 Wir möchten bemerken, dass wir in keiner Weise die Relevanz von freier Software in unkommerziellen, oft akademischen Kontexten als „Spielwiese" für Innovationen (Bresnahan 1998) in Abrede stellen.

14 Siehe etwa http://weblogs.mozillazine.org/mitchell/archives/2007/01/the_mozilla_foundation_achievi.html [6. Feb. 2008].

andere hingegen möchten den Bedürfnissen ihrer speziellen Zielgruppen gerecht werden und machen dem Projekt *Firefox* Konkurrenz – entweder, indem sie wie bei *Iceweasel* die Quelltexte des Browsers als Basis für ein abgeleitetes, aber verschiedenes Konkurrenzprojekt verwenden oder indem sie ein ganz eigenes Konkurrenzprodukt zu *Firefox* entwickeln.[15]

RHEL vs. CentOS Dass verschiedene Produkte auch bei großer Ähnlichkeit über Monopolmacht im oben genannten Sinne verfügen, wird am Beispiel der beiden Linux-Distributionen *Red Hat Enterprise Linux* (RHEL) und *CentOS* deutlich. Ersteres ist ein Produkt des kommerziellen Unternehmens *Red Hat, Inc.*, letzteres eine praktisch identische Version desselben Produkts, erzeugt aus den Quelltexten von RHEL mit dem Ziel, vollständige Kompatibilität zu RHEL zu erreichen. Dennoch haben beide Projekte sehr unterschiedliche Zielgruppen und Anwender, da der Kontext beider Projekte sehr verschieden ist. *Red Hat* verkauft Abonnements, Zertifizierungen und kommerziellen Support, während die CentOS-Community lediglich informelle Hilfe über Mailinglisten, Foren, Chat etc. anbietet. Obwohl also beide Projekte praktisch dieselbe Software vertreiben, sind sie für die Anwender nur sehr unvollständige Substitute füreinander, eben ob des sehr unterschiedlichen Kontexts.

Ähnliches gilt für die seit 2006 bestehende Initiative von *Oracle Corp.*, bei der die unter der GPL stehenden Quelltexte von RHEL unter dem Namen *Unbreakable Linux* zusammen mit entsprechenden Support-Verträgen vertrieben werden. Aber auch hier gilt, dass *Oracle* andere Kunden anspricht als *Red Hat*, schon alleine, weil *Red Hat* deutlich mehr Expertise bezüglich seiner Software hat als *Oracle*, die die Software lediglich unter einem anderen Namen weitervertreiben. *Oracle* spricht denn auch überwiegend Kunden an, die schon Kunden der Datenbank-Software von *Oracle* sind und denen vor allem an einem reibungslosen Zusammenspiel ihrer Datenbank und dem darunter liegenden Linux-System gelegen ist.

Neben den genannten auf RHEL basierenden Distributionen gibt es noch eine Unmenge weiterer Linux-Distributionen, alle mit unterschiedlichen Fokussen, Eigenheiten und Zielgruppen.[16] Hier wird deutlich, dass gerade weil der Preis als Variable im Wettbewerb von konkurrierenden Anbietern freier Software irrelevant, da Null ist, der Wettbewerb sich auf andere Bereiche, speziell die konkreten Eigenschaften der Software konzentriert. Auch hier zeigt sich wieder eine deutliche Analogie zu den Ausführungen von Chamberlin (1950): Wettbewerb findet nur selten über Preise statt, sondern vielmehr über das Produkt selbst, auch und gerade bei freier Software.

15 Siehe etwa den Konqueror-Browser des KDE-Desktop-Projekts unter http://www.konqueror.org [6. Feb. 2008].

16 Siehe etwa die unter http://distrowatch.com gelisteten Linux-Distributionen.

Die genannten Beispiele machen deutlich, dass freie Software *nicht* gleichbedeutend ist mit vollkommenem Wettbewerb, sondern eher dem oben beschriebenen Modell von monopolistischem Wettbewerb entpricht. Jede freie Software hat ihre über die eigentlichen Quelltexte hinausgehenden Eigenheiten und jedes Projekt Eigenschaften, die sich nur schwer nachbilden lassen. Damit hält jedes ein Monopol für ein mehr oder weniger eng gefasstes Marktsegment und steht als solches im Wettbewerb mit anderen Projekten, die ihrerseits Monopole halten. Jedes hat eine eigene strategische Ausrichtung, und jedes ist in einen unterschiedlichen Kontext eingebettet, sowohl auf Seiten der Produktion als auch auf Seiten der Anwendung.

5 Monopolelemente freier und proprietärer Software im Vergleich

Freie und proprietäre Software unterscheiden sich voneinander, insbesondere bezüglich der Stärke der Monopolmacht, die sie ausüben können. Freie-Software-Monopole sind gemeinhin deutlich schwächer als proprietäre (Gruber und Henkel 2006). Während es bei freier Software praktisch nur die Rechte am Namen der Software und die Kontrolle über eine Entwickler-Community sind, die ursächlich zum Schutz des Monopols genutzt werden können, kommt bei proprietärer Software noch die Kontrolle über die Quelltexte qua restriktiv ausgeübtem Urheberrecht und Geheimhaltung dazu, häufig in Verbindung mit Patenten für bestimmte Funktionalitäten der Software (Samuelson und Scotchmer 2002). Bei proprietärer Software ist es also für Konkurrenten wesentlich schwerer, identische oder direkt abgeleitete Produkte zu erstellen, ja überhaupt nur die Funktionalitäten oder bestimmte Standards einer Software nachzuahmen. Stattdessen gibt es aber auch einen höheren Anreiz, sinnvolle Alternativen zu bestehenden Monopolen zu entwickeln, diese genießen ja dann eben solchen Monopolschutz.[17]

Bei freier Software ist eine derartige Dominanz eines einzelnen Unternehmens nur schwer vorstellbar. Dafür ist es aber eben auch kaum möglich, mit freier Software direkt jene Monopolgewinne zu erwirtschaften, die für risikobehaftete Innovationen und professionelles Marketing gemeinhin notwendig sind. Gewinne können hier, trotz Monopol, oft nur über die oben schon angesprochenen Komplementäreffekte wie Anpassungen oder spezielle Erweiterungen der Software erfolgen, was aber nicht

17 Es gibt Fälle, in denen die Monopole bei proprietärer Software aufgrund von Netzwerk- und Skaleneffekten derart stark sind, dass es für Wettbewerber praktisch unmöglich ist, dem Monopol direkt Konkurrenz zu machen (Bresnahan 1998). Dann besteht die einzige sinnvolle Möglichkeit, das bestehende Monopol anzugreifen, darin, die Relevanz des Bereichs, in dem sich der Monopolist bewegt, selbst zu untergraben. Wenn also etwa webbasierte Software à la *Google* die heute noch dominante Desktop-Software à la *Microsoft Office* ablösen würde, dann käme es zu einer solchen von Bresnahan (1998) beschriebenen epochalen Verschiebung. *Microsoft* argumentiert derzeit tatsächlich, dass genau diese epochale Verschiebung nunmehr stattfindet (Rule et al. 2007).

immer praktikabel oder überhaupt möglich ist.[18] Diese Schwierigkeiten treffen mithin häufig genau auf jene Software zu, die am besten auf die Bedürfnisse von einfachen Benutzern zugeschnitten *wäre* – dort bräuchte es ja keine Anpassungen oder anderweitige Dienstleistungen mehr. Hier bliebe also nur die Finanzierung qua Subventionen, was angesichts fehlender Marktsignale allerdings nur ein bedingt taugliches Mittel der Ressourcensteuerung bleibt.[19]

6 Fazit

Freie Software weist durchaus Monopolelemente auf. Sie steht zwar per Definition unter einer Lizenz, die die völlig ungehinderte Weitergabe der Software mitsamt ihrer Quelltexte sichert, dennoch ist ein Freie-Software-Projekt ob der vorhandenen zentralisierten Kontrollmechanismen eine Firma im Sinne von Coase (1937): Es gibt ein Machtzentrum über die begrenzten Ressourcen für die Entwicklung der Software des Projekts sowie Namen und relevante Kontexte, woraus eine Marktmacht für das Software-Produkt selbst folgt.

Freie Software ist also nicht frei in einem unbedingten und absoluten Sinne, sie ist genauso an Namen, Personen und Kontexte gebunden wie proprietäre Software auch.

Die Monopolmacht eines Freie-Software-Projekts ist allerdings deutlich eingeschränkter als die eines proprietären Herstellers. Dies ist zwar aus einer statischen Perspektive günstig für die Anwender, nicht jedoch, wenn man die dynamischen Vorteile von Monopolen und der Aussicht auf solche berücksichtigt. Der oft gepriesene Vorteil freier Software – die Beschränkung von Monopolrenten wegen der Unmöglichkeit, Standards von nur einem Anbieter zu kontrollieren – ist denn gleichzeitig auch ihr größter Nachteil: Der Mehrwert landet zum überwiegenden Teil beim Konsumenten, kaum etwas verbleibt beim Produzenten. Nur wenn es seitens der Produzenten gelingt, aus den vorhandenen Kontexten Profit zu schlagen, lohnt es sich für diese, freie Software zu produzieren. Sonst bleibt freie Software auf Subventionen und Steuerfinanzierung angewiesen, was die Übermittlung von Präferenzen der Anwender hin zu den Produzenten und damit den sinnvollen und zielgerichteten Einsatz von Ressourcen ganz enorm erschwert.[20]

18 Wie schon in Fußnote 12 erwähnt: Ein Monopol muss nicht zwingend mit exzessiven Profiten einhergehen.

19 Es gibt populäre Paradigmen, denen zufolge alle unsere Erwägungen müßig sind, weil Menschen in vielen Fällen freiwillig und in einer neuartigen „sozialen" Art und Weise Werte schaffen, selbst und vor allem solche, die in Markt- oder Firmenkontexten schlicht nicht umsetzbar wären (von Hippel 2005*a*; Benkler 2006). Die ebenfalls populäre Steigerung dieses Modells verzichtet gar auf jegliche positive Anteile und konstruiert einen rein normativen „ethischen Imperativ", demzufolge es gilt, anderen durch proprietäre Software nicht „zu schaden". Wir halten solche Modelle, die empirisch nur dünn belegt und in ihren Implikationen letztlich eher utopisch als politisch sind, jedoch nicht auf die von uns betrachteten Softwaremärkte anwendbar.

20 Wir möchten am Rande anmerken, dass steuerfinanzierte und subventionierte Entwicklung von freier Software sich dort anbietet, wo die konkreten Funktionalitäten einer Software hinreichend spezifi-

Freie Software ist also nicht die institutionelle Lösung für sämtliche Gebiete der Softwareentwicklung, sondern wird auch in Zukunft überwiegend Bereiche bedienen, in denen Anpassungen und Serviceleistungen bezüglich einer Software strukturbedingt notwendig sind, zuvörderst bei Software mit Plattform-Charakter. Denn nur dort, wo bei freier Software aus Monopolmacht Profite generiert werden können, kann es auch zu Innovationen und Investitionen in Software-Produkte kommen. Dort wo dies nicht gelingt, bleiben die ökonomischen Anreize, gute Software-Produkte als freie Software zu bauen, unvollständig.

Literatur

Benkler, Y. (2006), *The Wealth of Networks: How Social Production Transforms Markets and Freedom*, Yale University Press. http://www.benkler.org/Benkler_Wealth_Of_Networks.pdf.

Bresnahan, T. (1998), New Modes of Competition and the Future Structure of the Computer Industry, *in* 'Competition, Convergence, and the Microsoft Monopoly', Bd. Progress and Freedom Foundation Volume, Kluwer. http://www.stanford.edu/~tbres/research/pff.pdf.

Bärwolff, M. (2006), 'Tight Prior Open Source Equilibrium', *First Monday* **11**(1). http://www.firstmonday.org/issues/issue11_1/barwolff/index.html.

Chamberlin, E. H. (1950), *The theory of monopolistic competition: A re-orientation of the theory of value*, 6. Aufl., Harvard University Press. Erstmals 1933 veröffentlicht.

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

Grand, S., von Krogh, G., Leonard, D. und Swap, W. (2004), 'Resource Allocation Beyond Firm Boundaries: A Multi-Level Model for Open Source Innovation', *Longe Range Planning* **37**, S. 591–610.

Gruber, M. und Henkel, J. (2006), 'New ventures based on open innovation—An empirical analysis of start-up firms in embedded Linux', *International Journal of Technology Management* **33**(4), S. 356–372.

Heylighen, F. (2007), Warum ist Open-Access-Entwicklung so erfolgreich? Stigmergische Organisation und die Ökonomie der Information, *in* B. Lutterbeck, M. Bärwolff und R. A. Gehring (Hrsg.), 'Open Source Jahrbuch 2007. Zwischen freier Softwareentwicklung und Gesellschaftsmodell', Lehmanns Media, Berlin, S. 165–180. http://www.opensourcejahrbuch.de.

Lancashire, D. (2001), 'Code, Culture and Cash: The Fading Altruism of Open Source Development', *First Monday* **6**(12). http://firstmonday.org/issues/issue6_12/lancashire/index.html.

ziert sind. Dies war etwa bei *Linux* der Fall, hier ging es primär um eine freie Implementierung des POSIX-Standards. Ebenso sind die GNU-Tools überwiegend Implementierungen von vorher bestehender Software. Völlig neue Produktinnovationen sind in diesem Bereich aber eben nicht zu erwarten.

Lerner, J. und Tirole, J. (2002), 'Some Simple Economics of Open Source', *Journal of Industrial Economics* **50**(2), S. 197–234. http://www.people.hbs.edu/jlerner/simple.pdf (Arbeitspapier aus dem Jahre 2000).

Lessig, L. (2002), Open Source Baselines: Compared to What?, *in* R. Hahn (Hrsg.), 'Government Policy toward Open Source Software', AEI-Brookings Joint Center for Regulatory Studies, Washington, DC, Kapitel 4, S. 50–68, notes at 99–105. http://www.aei.brookings.org/publications/abstract.php?pid=296.

Moglen, E. (2007), 'The Global Software Industry in Transformation: After GPLv3', http://www.archive.org/details/EbenMoglenLectureEdinburghJune2007StreamingVideo384kbits, transcript at http://www.archive.org/details/EbenMoglenLectureEdinburghJune2007text. Lecture in Edinburgh, Scotland, 26th June 2007.

O'Mahoney, S. und Ferraro, F. (2004), Managing the boundary of an 'open' project, IESE Research Papers D/537, IESE Business School. http://www.iese.edu/research/pdfs/DI-0537-E.pdf.

Pinker, R. (2006), 'From Gift Relationships to Quasi-markets: An Odyssey along the Policy Paths of Altruism and Egoism', *Social Policy and Administration* **40**(1), S. 10–25.

Priddat, B. P. (2006), Open Source als Produktion von Transformationsgütern, *in* B. Lutterbeck, M. Bärwolff und R. A. Gehring (Hrsg.), 'Open Source Jahrbuch 2006. Zwischen Softwareentwicklung und Gesellschaftsmodell', Lehmanns Media, Berlin, S. 109–121. http://www.opensourcejahrbuch.de.

Rule, C. F., Kanther, J. S., Smith, B. L., Snapp, M., Heiner, Jr., D. A., Holley, S. L. und Pepperman II, R. C. (2007), Memorandum of Points and Authorities of Microsoft Corporation in Opposition to Certain Plaintiff States' Motions to Extend the Final Judgments, Memorandum submitted to the US District Court of Columbia in civil action no. 98-1233 against Microsoft, Microsoft Corporation. http://www.microsoft.com/presspass/download/legal/settlementproceedings/11-07MemoranduminOpposition.pdf.

Samuelson, P. und Scotchmer, S. (2002), 'The Law and Economics of Reverse Engineering', *Yale Law Journal* **111**, S. 1575–1663. http://www.yalelawjournal.org/pdf/111-7/SamuelsonFINAL.pdf.

Wernerfeld, B. (1984), 'A resource-based view of the firm', *Strategic Management Journal* **5**(2), S. 171–80.

von Hippel, E. (2005*a*), *Democratizing Innovation*, MIT Press, Cambridge, MA. http://web.mit.edu/evhippel/www/democ.htm.

Unternehmen zwischen Offenheit und Profitstreben*

JOEL WEST

Der Ausdruck *open* wird schon seit langem von Unternehmen gebraucht (bzw. missbraucht), um den eigenen Kapitalanteil zu erhöhen. Das trifft sowohl auf Open Science, Open Standards, Open Source als auch auf Open Innovation zu, wobei für die jeweiligen *stakeholders* ganz verschiedene Aspekte dieser Unternehmensstrategie bedeutend sind und der Open-Source-Umfang des jeweiligen Unternehmens variiert. Häufig wird die Bedeutung des Ausdrucks verwischt und nicht richtig verstanden, da sich Unternehmen gern nur damit schmücken, um sich einen Wettbewerbsvorteil zu verschaffen. Im Folgenden werden unterschiedliche Formen und Gründe für die Wahl eines Open-Source-Geschäftsmodells dargestellt und ein Einblick gegeben, inwiefern Firmen dieses Prinzip für sich nutzen können, um Gewinne zu erzielen.

Schlüsselwörter: Open Innovation · Open-Source-Geschäftsmodell · Strategie

1 Einleitung

Ganz gleich, ob beim strategischen Management oder bei der *public policy*, Open–Source-Strategien entsprechen in den meisten Fällen einer der beiden folgenden Bedeutungen des Wortes *open*: Entweder handelt es sich um etwas, an dem jeder teilnehmen darf oder aber es ist etwas, das frei zugänglich und damit für jedermann einsehbar ist. Ein Beispiel für ersteres wäre die von Gabel (1987) untersuchte Open–Systems-Bewegung, während letzteres am ehesten Paul Davids Untersuchungen zu Open Science (vgl. David 1998, 2002, 2005) entspräche.

Das Adjektiv *open* wird noch in anderen Zusammenhängen verwendet, die nachfolgend erörtert werden sollen. Ein wichtiger Aspekt ist, dass *openness* ein Begriff ist, dessen Bedeutung sich aus dem Vergleich erschließt. So hat Grove zu Recht behauptet,

* Aus dem Englischen von Nadja Schüler.

dass in der Computerindustrie der 1990er Jahre mehr auf Open-Source-Basis gearbeitet wurde, als das im Zuge der Entwicklung von Großrechnern in den 1980er Jahren der Fall war. Die Computerindustrie der 1990er Jahre war wiederum weniger frei zugänglich, als das bei den Betriebssystemen und Prozessoren, wie sie derzeit in z. B. Mobiltelefonen zum Einsatz kommen, zu beobachten ist.[1]

Auf den ersten Blick scheinen diese Definitionen des Begriffs *openness* dem Ziel eines jeden Unternehmens, durch einen wie auch immer gearteten Wettbewerbsvorteil Gewinne zu erzielen, zu widersprechen. Wie es Porter (1985) formulierte, hängt ein solcher Vorteil von den ergriffenen Maßnahmen innerhalb der Wertschöpfungskette eines Unternehmens ab, wobei potenziell jede zur Kostenführerschaft oder Differenzierung führen kann. Offenbar führt Open Source an jedem beliebigen Punkt der Wertschöpfungskette dazu, dass die Aussicht auf einen Wettbewerbsvorteil in diesem Teil der betrieblichen Wertschöpfungskette vermindert – wenn nicht gar ausgeschlossen – wird.

Trotzdem hat das Interesse an Open-Source-Geschäftsmodellen als Wettbewerbsstrategie innerhalb der letzten 10 Jahre beträchtlich zugenommen und das sowohl was die Theorie betrifft als auch im Hinblick auf die größere Verbreitung derartiger Unternehmensstrategien. Ein Beispiel ist die zunehmende wirtschaftliche Bedeutung der Biomedizin, die sich auf die von David untersuchte Open Science stützt. Allerdings sind auch ganz neue Entwicklungen wie Open-Source-Software (DiBona et al. 1999) und Open Innovation (Chesbrough 2003*a*) zu verzeichnen, während die Globalisierung von Informationstechnologien weiterhin das Interesse an Open Standards (vgl. West 2006) gesteigert hat.

Ich werde mich auf systemorientierte Industriezweige wie die Computer- oder Mobilfunkindustrie beschränken, bei der verschiedene Firmen innerhalb einer Netzwerkwirtschaft zusammenarbeiten, um Gewinne zu erzielen und den Markt zu erobern. Typisch für die Unternehmen dieses Industriezweiges ist außerdem, dass sie sehr stark dem Mechanismus von Angebot und Nachfrage unterliegen sowie versuchen, ihren Marktanteil zu vergrößern und diesen gegenüber der Konkurrenz zu behaupten (Katz und Shapiro 1986; Arthur 1996).

Zunächst werde ich auf Open-Source- und kommerzielle Strategien eingehen, die dem Hersteller (Anbieter) in solchen Industriezweigen zur Verfügung stehen, um dann anschließend eine Einteilung der Gründe eines Unternehmens, sich für ein Open-Source-Geschäftsmodell zu entscheiden, vorzunehmen. Abschließend sollen Erkenntnisse und Anregungen gegeben werden, die als Grundlage für weitere Untersuchungen zu Open-Source-Unternehmen dienen sollen.

1 Für 80 bis 85 Prozent der CPUs von Mobiltelefonen werden Nutzungsgebühren in Höhe von etwa 10 Cent an *ARM Holdings Plc* gezahlt. Der Wettbewerb zwischen den einzelnen CPUs ist bei Mobiltelefonen sogar noch größer als bei CPUs für PCs.

2 Open Source vs. kommerziell – Unternehmensstrategien im Vergleich

Wie können Unternehmen in der auf Netzwerken basierenden Elektronikindustrie, wie der Computer-, Kommunikations- und Unterhaltungselektronik, Wettbewerbsvorteile erreichen? Während es Untersuchungen zu Firmen gibt, die sich durch die Anwendung von Open-Source-Strategien Wettbewerbsvorteile verschaffen (vgl. hierzu insbesondere Garud und Kumaraswamy 1993; West 2003), wird gemeinhin angenommen, dass Unternehmen nur durch proprietäre (d. h. firmenspezifische) Strategien, basierend auf geistigem Eigentum, Wettbewerbsvorteile erzielen können. In diesem Zusammenhang werde ich die proprietären Strategien entsprechend des heutigen Forschungsstandes darlegen und dann auf die davon abweichenden Open-Source-Strategien eingehen und aufzeigen, in welchem Umfang sie von Firmen genutzt werden.

2.1 Vorteile kommerzieller Strategien

Kommerzielle Strategien werden nicht nur seit langem angewandt, sie haben für Geschäftsführer auch von jeher eine gewisse Rechtsgültigkeit. Bei einem Markt, der kaum oder nur geringe Gewinnaussichten bietet, spricht man hinsichtlich des Wettbewerbs von einer Nullsummen-Situation, d. h., dass die Vergrößerung der Marktanteile einer Firma mit dem Verlust von Marktanteilen eines anderen Unternehmens einhergeht. Solche Nullsummen-Situationen beeinflussen die Beziehungen zwischen Käufer und Verkäufer, so wie es Porter (1980) in seinem Wirtschaftsmodell veranschaulicht hat: Wenn ein Unternehmen seine Konkurrenten gewissermaßen aus dem Weg räumt und somit den Wettbewerb verringert, dann nimmt (aufgrund fehlenden Ersatzes) der Einfluss der Kunden ab, während der Einfluss auf die Preisfestsetzung (und Gewinnspannen) seitens des Unternehmens zunimmt.

Dieses Nullsummen-Prinzip wird nirgends deutlicher als bei den durch die Vergabe von Patenten geschaffenen, zeitweiligen Monopolen. Patente sollen als Anreiz für neue Erfindungen dienen, da sie den Ausschluss von Konkurrenten vom Wettbewerb möglich machen.[2] Wie Teece (1986) gezeigt hat, steigt (oder fällt) der Wert eines Patents mit der Zeit, je nachdem, ob sich das Patent gegenüber der Konkurrenz bewährt (oder nicht).

Andere Formen des geistigen Eigentums – insbesondere Betriebsgeheimnisse und Urheberrecht – sind ebenfalls ein zentraler Bestandteil der Wettbewerbsstrategien proprietärer Softwareunternehmen. Die Budgets für Forschung und Entwicklung führender Unternehmen haben (von jeher) den Zugang zum Markt und die Nachahmung

2 Ein solcher Effekt ergibt sich nicht allein aus der Gewährung von geistigen Eigentumsrechten, denn andere Formen des Immaterialguts – insbesondere Betriebsgeheimnisse, Urheberrecht und Schutzmarken – ermöglichen davon unabhängige Erfindungen, weshalb die Gewinne durch geistiges Eigentum einer Firma sich nicht zwangsläufig auf den Wert des Immaterialguts eines anderen, konkurrierenden Unternehmens auswirken.

durch die Konkurrenz erschwert, was nur dann fruchten kann, wenn der Konkurrenz untersagt wird, die Technologie oder Produkte des Marktführers zu kopieren (Arthur 1996; Campbell-Kelly 2003; Cusumano 2004).

In umfangreichen Studien wurden sowohl für Softwarefirmen als auch für Hardwarefirmen Wege aufgezeigt, wie sie Wettbewerbsvorteile erreichen und behaupten können, indem sie proprietäre Standards und Plattformen entwickeln und überwachen (vgl. Morris und Ferguson 1993; Besen und Farrel 1994; Bresnahan und Greenstein 1999; West und Dedrick 2000; Gawer und Cusumano 2002). Unternehmen bieten zusätzlich eine Reihe von ergänzenden Produkten an, um ihre Plattform einerseits für den Kunden ansprechender gegenüber denen der Konkurrenz zu machen und andererseits, um die Switching-Kosten für die Kunden zu erhöhen und so den Kunden möglichst an sich zu binden.

2.2 Open-Source-Strategien in der Unternehmenspraxis

Untersuchungen haben ebenfalls ergeben, dass es mitunter für Firmen vorteilhaft sein kann, wenn sie Open-Source-Strategien anwenden. Die älteste Untersuchung hierzu wurde auf dem Gebiet der Open Standards gemacht (Gabel 1987; Garud und Kumaraswamy 1993; Grindley 1995). Im Gegensatz zu den oben erwähnten proprietären Unternehmensstrategien ermöglichen Open Standards den Wettbewerb unter Anbietern und werden daher von Benutzerorganisationen bevorzugt und unterstützt (Isaak 2006; West 2006). Dies kann von Open-Standards-Anbietern genutzt werden, um mit proprietären Anbietern in den Wettbewerb zu treten (Gabel 1987; Garud und Kumaraswamy 1993).

Open-Source-Software erfreut sich derzeit einem immer größer werdenden Interesse, wobei die Zahl der Firmen, die derlei Software herstellen, vertreiben und unterstützen, ständig wächst. Laut Definition bietet Open-Source-Software (OSS) Anwendern und potenziellen Konkurrenten eine Reihe von ganz bestimmten Rechten hinsichtlich der Nutzung dieser Software (DiBona et al. 1999). Unternehmen haben daher diverse Unternehmensstrategien entwickelt, die die Verfügbarkeit von Open-Source-Inhalten erhöhen und OSS für den Kunden attraktiver machen sollen (West 2003; Välimäki 2005). Allerdings unterscheiden sich diese Strategien im Open–Source-Umfang gegenüber potentiellen Konkurrenten erheblich, je nachdem, welchen Eigentumsrechten und Lizenzen die Software unterliegt (West 2007).

Ein weiteres Beispiel von Open-Source-Unternehmensstrategien ist das Open–Innovation-Prinzip wie es Chesbrough beschrieben hat (Chesbrough 2003*b*; Chesbrough et al. 2006). Einige dieser Strategien wirken sich tendenziell negativ auf die Open-Source-Wirtschaft aus. So können Firmen aus ihrem patentierten Immaterialgut durch Lizenzvergaben Gewinn schlagen, was denjenigen, die kein geistiges Eigentum besitzen, den Zugang zum Markt ermöglicht und bei bestehenden (oder potenziellen)

Konkurrenten höhere Kosten verursacht.[3] Eine Umstellung auf an solche Nutzungsgebühren gebundene Lizenzen kann für neue Entwicklungen ganz besonders dann hinderlich sein, wenn sie von Einrichtungen (wie z. B. Universitäten) übernommen wird, die ihr Wissen zuvor der Wirtschaft oder anderen auf diesem Gebiet tätigen Forschern frei zugänglich gemacht haben (Fabrizio 2006).

Gleichzeitig wird durch die Verlagerung eines Open-Innovation-Netzwerks, wenn es über die Unternehmensgrenzen hinaus zugänglich gemacht wird, eine ganze Branche oder ein ganzes Marktsegment frei zugänglich, da die vertikale Integration als Hindernis für unbeteiligte Firmen an Bedeutung verliert. Außerdem subsumieren die Open-Innovation-Strategien, die sich nach Chesbrough (2003*a*) so genannte *innovator benefactors* zu Nutze machen, das Kollektivgütermodell, das David (1998) für Open Science entwickelt hat.

Weitere Aspekte von Open Innovation ergeben sich aus dem Nutzerinnovationsmodell nach von Hippel (1988, 2005) sowie aus dem kumulativen Innovationsprozess, wie er von Scotchmer (2004) aufgezeigt wurde, welche beide sowohl die Mitwirkung der Kunden als auch der Konkurrenten erfordern. Durch beide Verfahrensweisen wird eine Branche oder ein Marktsegment „offener".

2.3 Open Source – In welchem Umfang?

Patente unterscheiden sich hinsichtlich der Barrieren, die sie für die Konkurrenz schaffen. Oft sind diese abhängig von der Art der Erfindung und den Gesetzen des jeweiligen Landes. Ferner unterschieden sich Unternehmen hinsichtlich der beabsichtigten Nutzung des Patents. Ausgehend von einer im Jahre 1994 von amerikanischen Herstellern durchgeführten Studie hat Cohen et al. (2000) die folgenden wesentliche Gründe für das Patentieren von Neuentwicklungen herausgearbeitet:

1. Nachahmungen verhindern
2. dem Patentieren von ähnlichen Erfindungen vorbeugen
3. Nachfolgemodelle ausschließen
4. den Verhandlungsvorteil nutzen
5. den guten Ruf fördern
6. Lizenzeinnahmen machen

3 Ein bezeichnendes Beispiel hierfür ist das Patent von *Qualcomm* für die Entwicklung des CDMA–Verfahrens in der Mobilfunkbranche. Allerdings haben die Lizenzierung des Immaterialguts und der Verkauf von Schlüssellösungen durch *Qualcomm* neuen bzw. kleineren Herstellern wie *Samsung*, *LG* und *Sanyo*, die selbst über keine Forschungs- und Entwicklungsabteilung verfügen, erst den Marktzutritt ermöglicht. Verständlicherweise haben sich die etablierten Mobiltelefonanbieter (wie *Nokia* und *Broadcom*) aber gesträubt, die Nutzungsgebühren für das Patent zu zahlen, die für sie die Qualcomm-Steuer war.

Die ersten beiden Absichten verringern für Konkurrenten die Aussicht auf Zugang zum entsprechenden Technologiesegment, während die anderen Gründe für sich genommen nicht unbedingt den Zugang zum Markt behindern.

Die Exklusivität von Patenten hat in Verbindung mit einer Zunahme an Patenteintragungen im amerikanischen Softwaresektor dazu geführt, dass der Kampf gegen die immer populärer werdende Open-Source-Software wahrscheinlicher wird.[4] Dies hat zwei neue, weniger restriktive Patentstrategien hervorgebracht. Eine davon ist die Zusicherung der freien Nutzung von unternehmenseigenen Patenten, unter der Bedingung, dass der geschützte Code unter eine Open-Source-Lizenz gestellt wird. Diesen Ansatz verfolgte *IBM* im Jahr 2005, um eine weitere Verbreitung von Linux zu fördern.[5] Weiterhin bemühen sich Open-Source-Anhänger, den genauen Stand der Technik zu ermitteln, um so Patente, die nie hätten genehmigt werden sollen, zur Löschung zu bringen.

Open Standards variieren zudem erheblich in puncto Restriktivität. Einige Verfahrensweisen zur Schaffung neuer Standards sind in Bezug auf die Mitwirkung verschiedener *stakeholders* an sich weniger restriktiv bzw. kommerziell als andere und stellen somit eher ein gemeinsames als ein persönliches Ziel dar (Krechmer 2006). Bei der Standardisierung werden sowohl der Open-Source-Umfang des Verfahrens als auch das Ergebnis je nach *stakeholder* sehr stark variieren (West 2006).

3 Gründe für die Wahl eines Open-Source-Geschäftsmodells

Hier sollen die Beweggründe eines Unternehmens, auf Open-Source-Basis zu arbeiten, entsprechend vorliegender Forschungsergebnisse auf diesem Gebiet in sechs verschiedene Kategorien eingeteilt werden. Als Parameter dient der Hauptgrund eines Unternehmens, eine Open-Source-Strategie zu verfolgen und damit das, was sich eine Firma davon verspricht, nicht kommerziell zu sein. Nichtsdestotrotz wird die genaue Unterteilung der einzelnen Kategorien mitunter nicht ganz eindeutig sein, wenn die entsprechenden Unternehmensstrategien, Ursprünge eines Wettbewerbsvorteils oder die wirtschaftlichen Kräfte gegenübergestellt werden. Die Beziehungen, die zwischen den Firmen innerhalb eines nicht kommerziellen Wertenetzwerks bestehen, mögen sich sehr ähneln, ganz gleich, ob Open Source von außen aufgezwungen wurde, gemeinsam verbreitet oder aber von einem einzelnen Unternehmen aus eigenem Interesse angeregt wurde.

4 Open-Source-Software lässt sich am besten von einer virtuellen gemeinnützigen Gemeinschaft wie der *Apache Software Foundation* verwalten. Dennoch gewinnt auch das Sponsoring bei Open-Source-Software immer mehr an Bedeutung (West und O'Mahony 2007). Solche Software wird im Allgemeinen von Unternehmen verwaltet, weshalb sie in Bezug auf Patente den gleichen Bedingungen wie firmeneigene proprietäre Software unterliegt.

5 Siehe http://www.heise.de/newsticker/meldung/54971 und http://www.ibm.com/ibm/licensing/patents/pledgedpatents.pdf.

3.1 Exogene Faktoren

In einigen Fällen bleibt den Anbietern einer Branche gar nichts anderes übrig, als ein Open-Source-Geschäftsmodell anzunehmen. Hierfür gibt es unterschiedliche Gründe:

Die Notwendigkeit für Open Source kann sich für einen ganzen Industriesektor ergeben. Einer dieser Gründe sind begrenzte finanzielle Mittel, die eine auf Wettbewerbsvorteile und Gewinn gerichtete Strategie notwendig machen (Teece 1986). Ein weiteres Beispiel ist die von der Regierung festgelegte Open-Source-Politik, wie sie im Jahr 1950 zur Anwendung kam, als die amerikanischen Patentgesetze *AT&T* sowie *RCA* dazu zwangen, die Lizenzen an ihren Patenten an alle Konkurrenten zu vergeben.

Auch die Struktur einer bestimmten Branche kann Open Source unumgänglich machen. Bis in die 1990er Jahre hatten die meisten europäischen Telekommunikationsoperatoren ein nationales Monopol inne und so beherrschten einige wenige Großunternehmen den Markt, sie waren oligopole Käufer von Telekommunikationsequipment und legten die Bedingungen für die potenziellen Anbieter fest. Doch selbst, wenn es sich nicht um Oligopole handelt, können Großkunden Open Source als Lieferbedingung einfordern, so wie es die US-Regierung mit den Open Systems in den 1990er Jahren gemacht hat (Isaak 2006). Oder aber es kommt zur Entstehung eines Marktes, bei dem Kunden neue proprietäre Produkte zugunsten altbekannter Open-Source-Lösungen ablehnen.

3.2 Endogene Faktoren

Einige Unternehmen können sich für Open Source entscheiden, ohne dass ihnen dies bewiesenermaßen einen Wettbewerbsvorteil verschafft oder einen andersgearteten Vorteil zur Folge hat. Ein solches Management geht davon aus, dass Open Source in jedem Fall vorteilhaft ist – ganz gleich, ob für die Aktieninhaber oder *stakeholders* – und dass dies den Verzicht auf Exklusivität seitens des Unternehmens rechtfertigt.

Man könnte meinen, dass dies nur dort zutrifft, wo das Unternehmen die Werte des Geschäftsführers repräsentiert, so wie das bei Firmen (wie *Ben & Jerry's* oder *The Body Shop*) der Fall ist, die Sozialrecht oder andere Werte ihrer Gründer verkörpern. Das beste Beispiel einer solchen ideologisch ausgerichteten Firmenstrategie in der IT-Branche wären Open-Source-Software-Startups – einer Firmengruppe, bei der die Unternehmer die Werte der Open-Source-Community widerspiegeln und sich der Open-Source-Definition verschreiben.

Dennoch ist diese Form von Open Source bei den wichtigsten Open-Source-Startups, die von Risikokapitalgebern unterstützt werden, nicht vorzufinden. D. h., dass die Hauptanleger (Risikokapitalgeber) in Open Source einen Widerspruch zu den eigenen Interessen sehen und deshalb davon abrücken, ganz gleich, welche Werte von den Un-

ternehmern suggeriert werden.[6] Die Entscheidung der Unternehmen *Ben & Jerry's* und *The Body Shop*, die Firmengründer und deren Werte infolge schwacher Umsätze zugunsten der Aktienteilhaber in den Hintergrund treten zu lassen, entspricht genau diesem Prinzip.

4 Das Open-Source-Geschäftsmodell als Strategie für Markteinsteiger

Open Source wird häufig von Herausforderern oder Markteinsteigern genutzt, um mit einem etablierten proprietären Konkurrenzunternehmen in den Wettbewerb treten zu können (Grindley 1995; West 2006). Solche Marktherausforderer würden ihr Unternehmen selbst gern als proprietäres führen und dementsprechende Gewinne erzielen, nutzen die Open-Source-Strategie jedoch, weil es der einzige Weg ist, sich überhaupt Zugang zum Markt zu verschaffen oder mit der Konkurrenz mitzuhalten. Shapiro und Varian beschreiben diesen wichtigen Kompromiss in ihrem Buch:

> „Do you choose an *open* approach by offering to make the necessary interfaces and specifications available to others, or do you attempt to maintain control by keeping your system proprietary?
>
> Proprietary control will be exceedingly valuable if your product or system takes off. [. . .] [A]n installed base is more valuable if you do not face rivals who can offer products to locked-in customers.
>
> [. . .] Failure to open up a technology can spell its demise, if consumers fear lock-in or if you face a strong rival whose system offers comparable performance but is not proprietary. [. . .] Openness will bolster your chances of success by attracting allies and assuring would-be customers that they will be able to turn to multiple suppliers down the road.
>
> Which route is best, openness or control? The answer depends on whether you are strong enough to ignite positive feedback on your own."

5 Open-Source-Geschäftsmodelle zur Kommodisierung

In einigen Fällen werden Firmen bereitwillig die partielle Kommodisierung ihres Nutzungsversprechens in Kauf nehmen, um die Gewinne in anderen Sparten zu sichern. Diese Strategie ist sehr verbreitet und wird gemeinhin als „Rasierklingen-Geschäftsmodell" bezeichnet. Während Rasierer einst verschenkt bzw. zu Schleuderpreisen angeboten wurden, um den Verkauf von Rasierklingen anzukurbeln, gibt es heutzutage noch ganz andere Gründe für die Anwendung einer solchen Unternehmensstrategie.

6 Im Jahr 2006 verklagten die Investoren von *Medsphere Systems Corp.*, einem Open-Source-Startup, deren Gründer, da sie nach ihrem Rausschmiss den Firmencode in einer Open-Source-Lizenz offen gelegt hatten.

Unternehmen, die im Mehrpunktwettbewerb stehen, versuchen beispielsweise, durch Verluste von Marktanteilen seitens der Konkurrenten die eigene Position zu Lasten der Konkurrenz auszubauen oder diese gar vollends aus dem Weg zu schaffen. Des Weiteren gibt es Firmen, die versuchen, durch partielle Kommodisierung ihres Nutzungsversprechens die Preise selbst zu bestimmen und die Gewinne zu sichern, so wie das von *Microsoft* und *Intel* praktiziert wurde, die den Wettbewerb zwar unter den Anbietern von PC-Systemen ankurbelten, jedoch nicht unter den Anbietern des entsprechenden Zubehörs.

Ein typisches Beispiel für diese Form der Kommodisierung wurde von West (2007) bei Computer-Architekturen festgestellt und als *commoditizing up the stack* bezeichnet. Bei technischem Zubehör eines Informationssystems fördern Anbieter wie *Oracle* oder *SAP* die Kommodisierung von Open-Source-Software im unteren Bereich der Computer-Architektur, während sie die Kommodisierung ihrer eigenen Zubehörteile in diesem bestimmten Bereich unterbinden. Auf diese Weise hoffen sie, die Gesamtkosten für den Käufer zu senken, ohne dabei gleichzeitig die eigenen Preise und Gewinne zu drücken. Dieser Nullsummenwettbewerb um Einnahmen und Gewinne unter den Anbietern wird als Maximierung des *wallet share*[7] bezeichnet.

6 Wenn Open Source gar nicht Open Source ist

Oftmals sind die selbsternannten Open-Source-Strategien der Unternehmen viel kommerzieller, als es der Name vermuten lässt. In einigen Fällen ist das eine ziemliche Unverschämtheit der Unternehmen, wie etwa bei *Digital Equipment Corp.*, die ihr proprietäres Betriebssystem kurzerhand in *OpenVMS* umbenannt haben.

Häufig wird die Bezeichnung *open* verwendet, um für Kunden attraktiv zu sein und gleichzeitig der Konkurrenz den Zugang zum Markt zu erschweren und somit einen richtigen Wettbewerb zu unterbinden.

Auf dem Gebiet der Standards war der gesamteuropäische Mobilfunkstandard GSM einer der bekanntesten und erfolgreichsten der 1990er Jahre. Dieser Standard wurde in Anlehnung die Open-Source-Definition von Krechmer (2006) entwickelt. Dennoch basierte der Standard auf einer Reihe von patentierten Schlüsselstandards, an die bestimmte Nutzungsgebühren gebunden waren, und diese Nutzungsgebühren (geschätzte 15 US-Dollar pro Handset) schlossen logischerweise die meisten der potenziellen Markteinsteiger aus (Bekkers 2001; West 2006).

Bei einigen Standards stellen nicht Patente, sondern die Festlegung und Freigabe von Schlüsselspezifikationen eine Hürde für die Konkurrenz dar. Wie von MacKie-Mason und Netz (2006) beschrieben, hat *Intel* zwei PC-gebundene Standards (*AGP* und *USB 2.0*) entwickelt und deren Ausführungsbeschreibungen für Anwender offengelegt. Die Freigabe der Standardspezifikationen, die für die Konkurrenz notwendig sind, um

7 *Wallet Share* bezeichnet den Anteil des Unternehmens am spezifischen Einkaufsvolumen des Kunden.

den Standard zu nutzen, wurde jedoch so lange hinausgezögert, bis die Intel-eigenen Produkte auf dem Markt waren.

Solche Strategien, die Anwendern freien Zugang ermöglichen, aber gleichzeitig die Konkurrenz außen vor lassen, sind unter den Open-Source-Strategien sehr verbreitet. Unternehmen, die mit solchen „dualen Lizenzen" arbeiten, schließen die Konkurrenz von vornherein aus und verhindern als Inhaber des Software-Urheberrechts den Preiswettbewerb (Välimäki 2005). Andere Firmen wiederum machen nur bestimmte Teile ihrer technischen Entwicklungen frei zugänglich und behalten es sich vor, Schlüsseltechnologien nur proprietär anzubieten, so wie es *Apple* mit seinem Betriebssystem und seinem Browser gemacht hat. Diese Verfahrensweise wird von West (2003) als *opening parts* (partielle Entkommerzialisierung) bezeichnet. Letztlich gibt es noch einige Unternehmen, die einen überwiegenden Teil ihrer Produkte und Leistungen auf Open-Source-Basis anbieten und nur einen kleinen Teil proprietär auf den Markt bringen, so wie es von *Red Hat* praktiziert wird, um seine Kunden zu binden und sich so einen entsprechenden Marktanteil zu sichern (West 2007).

Im Allgemeinen sind selbst bei *value networks*, die sehr stark auf Open-Source–Strategien basieren, die Gewinnaussichten für die einzelnen Bereiche unterschiedlich hoch. Iansiti und Levien (2004) haben bei ihrer intensiven Beschäftigung mit *value networks* herausgearbeitet, dass die Beteiligten dieser ein gemeinsames Interesse an deren Erhalt haben. D. h. dass bei diesen Systemen der Hauptakzent auf der gemeinsam erzielten Wertschöpfung liegt und es weniger darauf ankommt, im Alleingang Gewinne zu erzielen. Das heißt aber auch, dass bestimmte Rollen innerhalb dieses Netzwerks ausgesprochen wichtig sind, weil sie von einzelnen Unternehmen besetzt werden. Diese Tatsache wiederum bestärkt die Unternehmen darin, ihren Einfluss, der ihnen durch ihre Rolle innerhalb des Netzwerks zukommt, nicht aufzugeben.

7 Zur Markterweiterung

In den übrigen Fällen gehen die Firmen davon aus, dass durch Open Source der Markt für eine bestimmte Technologie vergrößert werden kann, weshalb die Verluste, die sich durch die Gewinnbeteiligung der Konkurrenten ergeben, zu vernachlässigen sind. Man spricht von „der Markterweiterung" statt „die bestehenden Marktanteile untereinander aufzuteilen". Oder anders formuliert: „10 Prozent von irgendetwas sind immer noch mehr als 100 Prozent von nichts". Im IT-Bereich wird Open Source dazu eingesetzt, kritische Anbieter und Anwender für sich zu gewinnen und die Annahme einer bestimmten Produktkategorie zu fördern und deren Wachstum anzukurbeln. Eine solche Herangehensweise käme bei anderen Produktkategorien oder Konsumgütern wohl kaum in Frage, lässt sich aber im IT-Bereich aus zwei Gründen problemlos anwenden: Erstens besteht in Branchen, in denen sich die Technik schnell weiterentwickelt, immer die Gefahr, dass eine Technologie veraltet bzw. von der nächsten Generation abgelöst wird.

Zweitens unterliegen diese Branchen den Netzwerkeffekten, weshalb es einer kritischen Masse bedarf, um eine positive Netzwerkexternalität, einen so genannten *feedback-loop*[8], zu erzielen. Solche Rückkopplungseffekte ergeben sich sowohl für konkurrenzlose als auch für konkurrierende Technologien, bei denen der Kunde zwischen verschiedenen Anbietern wählen kann, wie z. B. bei EC-Automaten oder lokalen Netzwerken (Saloner und Shepard 1995; von Burg 2001).

Laut Simcoe (2006) ist die richtige Kombination von Open Source und proprietären Geschäftsstrategien für bereits entwickelte Technologien das beste Rezept. Hierbei sollte man beachten, dass weder das eine noch das andere Geschäftsmodell überwiegt, denn sich zu sehr auf Open Source zu stützen, würde die Gewinnerzielung erschweren, während ein Verzicht auf Open Source bei der Wertschöpfung hinderlich sein würde.

8 Diskussion

Die vorgenommene Einteilung von Open-Source-Strategien kann auf drei wesentliche Gründe reduziert werden: Entweder haben Firmen schlicht und einfach keine andere Wahl, als dieses Geschäftsmodell anzunehmen oder sie nutzen Open Source gezielt, um der Konkurrenz zu schaden oder aber die durch Open Source erzielten Gewinne übersteigen jedweden Verlust.

Während die ersten beiden Formen (Open Source als einzige Möglichkeit für den Marktzutritt sowie Open Source zur Schwächung der Konkurrenz) weiterer Untersuchungen bedürfen, soll im Folgenden die „einzig wahre" Open-Source-Strategie im Mittelpunkt stehen. Unter welchen Bedingungen ist diese Form der Geschäftsführung die richtige Strategie für ein Unternehmen? Da es Firmen, die auf Open-Source-Basis arbeiten, nur unter ganz bestimmten Bedingungen gelingt, im Gegenzug zu ihren kostenfrei angebotenen Produkten ihren Marktanteil zu vergrößern, ist die Anwendung dieses Geschäftsmodells sehr selten. Der Erfolg eines solchen Geschäftsmodells ist zudem abhängig vom Vorhandensein (bzw. der Entwicklung) einer Positivsummenbeziehung unter den einzelnen Konkurrenten. Ein Beweis hierfür wäre die Tatsache, dass Open Source in großem Stil meist nur in Branchen vorzufinden ist, die von mehreren Unternehmen zugleich dominiert werden, wodurch keine der Firmen darauf setzen kann, alle Gewinne für sich allein zu beanspruchen. Dennoch wurden Patente in letzter Zeit genutzt, um die Interessen mehrerer Anbieter zu vereinen, ohne zwangsläufig allen Firmen die Wettbewerbsteilnahme zu ermöglichen, wie das bei *MPEG LA* und deren Patent für den Videostandard *MPEG-4* oder beim Streit um die Videostandards *HD*, *DVD* und *Blu Ray* der Fall war.

8 Engl. für Rückkopplungsschlaufe – Je mehr Anwendungsprogramme für ein Betriebssystem zur Verfügung stehen, desto höher ist der aus den vielfältigen Anwendungsmöglichkeiten entstehende Nutzen für den Kunden und das Betriebssystem wird verstärkt nachgefragt. Je häufiger ein Betriebssystem nachgefragt wird, desto lohnender ist es umgekehrt, für das besonders häufig abgesetzte Betriebssystem Anwendungsprogramme zu schreiben, um sich einen großen Markt zu sichern.

Eine weitere Annahme wäre, dass die bisherige Wirtschaftsstruktur mit ihren Normen und Bedingungen zu einer umfangreicheren Anwendung von Open Source führt. Doch auch diese ist ein Irrtum, für den es entsprechende Beispiele gibt. Einerseits hat die *Internet Engineering Task Force* (IETF) seit mehr als 20 Jahren Internettechnologien auf Open-Source-Basis entwickelt. Andererseits hat die zunehmende Globalisierung dazu geführt, dass die Zusammenarbeit der Unternehmen auf dem europäischen Telekommunikationssektor in den 1960ern und 1970ern von Patentanmeldungen für Mobilfunkstandards wie *2G* (*GSM*) und *3G* (*UMTS*) abgelöst wurde.

Weitere Untersuchungen setzen die Bedeutung der User Innovation und der kumulativen Innovation (wie sie von von Hippel und Scotchmer beschrieben wurden) mit Open Source in Beziehung.

Ist diese Innovation der Vorgänger von Open Source, deren Folge oder steht sie gar in keinerlei Zusammenhang? Wissenschaftler wie Fabrizio und David haben die Privatisierung von Open Science beklagt. Kann Open Source für das Unternehmen (und nicht nur für die Kunden) rentabel sein?

Vorhersagen dieser Art erfordern genauere Formen und Einteilungen von Open Source. So haben Laursen und Salter (2006) beispielsweise vorgeschlagen, die von Chesbrough (2003*a*) beschriebene Open Innovation messbar zu machen. Die Einteilung von Open Standards nach Krechmer (2006) ist sicherlich ein Anfang, doch Ähnliches steht für andere Gebiete (sowie für die Open Innovation im weiteren Sinne) noch aus.

Literatur

Arthur, W. B. (1996), 'Increasing Returns and the New World of Business', *Harvard Business Review* **74**(4).

Bekkers, R. (2001), *Mobile telecommunications standards: GSM, UMTS, TETRA, and ERMES*, Artech House, Boston, Mass.

Besen, S. M. und Farrel, J. (1994), 'Choosing How to Compete: Strategies and Tactics in Standardization', *Journal of Economic Perspectives* **8**(2).

Bresnahan, T. F. und Greenstein, S. (1999), 'Technological competition and the structure of the computer industry', *Journal of Industrial Economics* **47**(1).

Campbell-Kelly, M. (2003), *From airline reservations to Sonic the Hedgehog: a history of the software industry*, MIT Press, Cambridge, Mass.

Chesbrough, H. (2003*a*), 'The Era of Open Innovation', *Sloan Management Review* **44**(3).

Chesbrough, H. (2003*b*), *Open Innovation: The New Imperative for Creating and Profiting from Technology*, Harvard Business School Press, Boston, Mass.

Chesbrough, H., Vanhaverbeke, W. und West, J. (2006), *Open Innovation: Researching a New Paradigm*, Oxford University Press, Oxford.

Cohen, W. M., Nelson, R. R. und Walsh, J. P. (2000), Protecting Their Intellectual Assets: Appropriability Conditions and Why U.S. Manufacturing Firms Patent (or Not), Working paper, National Bureau of Economic Research.

Cusumano, M. A. (2004), *The Business of Software*, Free Press, New York.

David, P. (1998), 'Common Agency Contracting and the Emergence of Open Science Institutions', *American Economic Review* **88**(2).

David, P. (2002), 'The Economic Logic of Open Science and the Balance between Private Property Rights and the Public Domain in Scientific Data and Information: A Primer', *Stanford Institution for Economic Policy Research, Discussion Paper* (02-30).

David, P. (2005), Can Open Science be Protected from the Evolving Regime of IPR Protections?, Working Paper 0502010, EconWPA. http://ideas.repec.org/p/wpa/wuwpio/0502010.html.

DiBona, C., Ockman, S. und Stone, M. (1999), *Open Sources: Voices from the Open Source Revolution*, O'Reilly, Sebastopol, CA.

Fabrizio, K. (2006), The Use of University Research in Firm Innovation, *in* W. V. Henry Chesbrough und J. West (Hrsg.), 'Open Innovation: Researching a New Paradigm', Oxford University Press, Oxford, S. 134–160.

Gabel, H. L. (1987), Open Standards in Computers: The Case of X/OPEN, North Holland, Amsterdam.

Garud, R. und Kumaraswamy, A. (1993), 'Changing competitive dynamics in network industries: An exploration of Sun Microsystems open systems strategy', *Strategic Management Journal* .

Gawer, A. und Cusumano, M. A. (2002), *Platform Leadership: How Intel, Microsoft, and Cisco Drive Industry Innovation*, Harvard Business School Press.

Grindley, P. (1995), *Standards, strategy, and policy: cases and stories*, Oxford University Press.

Iansiti, M. und Levien, R. (2004), *The Keystone Advantage: What The New Dynamics of Business Ecosystems Mean for Strategy, Innovation, and Sustainability*, Harvard Business School Press.

Isaak, J. (2006), 'The Role of Individuals and Social Capital in POSIX Standardization', *International Journal of IT Standards & Standardization Research* .

Katz, M. L. und Shapiro, C. (1986), 'Technology Adoption in the Presence of Network Externalities', *Journal of Political Economy* .

Krechmer, K. (2006), 'Open Standards Requirements', *International Journal of IT Standards and Standardization Research* .

Laursen, K. und Salter, A. J. (2006), 'Open for Innovation: The role of openness in explaining innovation performance among UK manufacturing firms', *Strategic Management Journal* .

MacKie-Mason, J. S. und Netz, J. S. (2006), Manipulating interface standards as an anti-competitive strategy, *in* S. Greenstein und V. Stango (Hrsg.), 'Standards and Public Policy', S. 231–259.

Morris, C. R. und Ferguson, C. H. (1993), 'How Architecture Wins Technology Wars', *Harvard Business Review* .

Porter, M. (1980), *Competitive Strategy*, Free Press.

Porter, M. (1985), *Competitive Advantage*, Free Press.

Saloner, G. und Shepard, A. (1995), 'Adoption of technologies with network effects: An empirical examination of the adoption of automated teller machines', *Rand Journal of Economics* .

Scotchmer, S. (2004), *Innovation and incentives*, MIT Press, Cambridge, Mass.

Simcoe, T. (2006), Open Standards and Intellectual Property Rights, *in* H. Chesbrough, W. Vanhaverbeke und J. West (Hrsg.), 'Open Innovation: Researching a New Paradigm', S. 161–183.

Teece, D. (1986), 'Profiting from technological innovation: Implications for integration, collaboration, licensing and public policy', *Research Policy* **15**(6), S. 285–305.

Välimäki, M. (2005), *The Rise of Open Source Licensing: A Challenge to the Use of Intellectual Property in the Software*, Turre.

West, J. (2003), 'How Open is Open Enough? Melding Proprietary and Open Source Platform Strategies', *Research Policy* .

West, J. (2006), The Economic Realities of Open Standards: Black, White and Many Shades of Gray, *in* S. Greenstein und V. Stango (Hrsg.), 'Standards and Public Policy', S. 87–122.

West, J. (2007), Value Capture and Value Networks in Open Source Vendor Strategies, *in* 'Proceedings of the 40th Annual Hawaii International Conference on System Sciences'.

West, J. und Dedrick, J. (2000), 'Innovation and Control in Standards Architectures: The Rise and Fall of Japan's PC-98', *Information Systems Research* .

West, J. und O'Mahony, S. (2007), Sponsored Open Source Communities: a Vehicle for Open Innovation?, *in* 'European Academy of Management 2007 conference'.

von Burg, U. (2001), *The triumph of Ethernet: technological communities and the battle for the LAN standard*, Stanford University Press.

von Hippel, E. (1988), *The Sources of Innovation*, Oxford University Press.

von Hippel, E. (2005), *Democratizing Innovation*, MIT Press.

Open-Source-Marketing – Warum Konsumenten am Marketing von Unternehmen teilnehmen

KLAUS-PETER WIEDMANN, LARS PANKALLA
UND SASCHA LANGNER

Wenn Konsumenten sich freiwillig in die Vermarktungs- und Absatzprozesse von Unternehmen einbringen oder aus eigenem Antrieb Marketing für Dienstleistungen bzw. Produkte betreiben, dann ist das ein auffallender Umstand. Dieses Phänomen kann man beim Open-Source-orientierten Marketing beobachten. Einst als Pendant zur klassischen Softwareentwicklung ins Leben gerufen, existieren OS-Netzwerke (OSN) heute auch in unterschiedlichsten Gebieten: z. B. als Zusammenschlüsse von Kreativen, als Bildungsnetzwerke oder als kollaborative Marketing-Communitys. Doch aufgrund der Freiwilligkeit, der i. d. R. nicht monetären Entlohnung und des weitgehenden Verzichts der teilnehmenden Konsumenten auf private Eigentumsrechte stellt Open-Source-Marketing (OSM) ein ökonomisches Paradox dar, da offensichtlich nur die Unternehmen von dieser Kooperationsart profitieren (bspw. Kosten einsparen). Folglich stellt sich die Frage: Was treibt Konsumenten zum freiwilligen Marketing an? Auf Grundlage relevanter Motivationstheorien und aktueller Forschungsergebnisse liefert dieser Beitrag eine empirische Analyse zur Identifizierung der wesentlichen konsumentenseitigen Motivationsdimensionen und -Faktoren und ist damit ein erster Schritt in Richtung der Erklärung OS-orientierter Marketing-Communitys.

Schlüsselwörter: Open-Source-Marketing · Motivation

1 Einleitung

Traditionelle Managementansätze drohen, künftig immer weniger zu greifen. Vor allem die konventionelle Kommunikationsmethodik stößt zunehmend auf Probleme, wenn es darum geht, Botschaften streuverlustfrei an relevante Zielgruppen heranzutragen

(vgl. Spitzer und Swidler 2003; Rose 2004). Bei fortschreitender Werbedichte bewirkt diese vermehrte Reizüberflutung und Informationsüberlastung bei den Konsumenten latentes Misstrauen und mitunter sogar Reaktanzreaktionen (vgl. Oetting 2006).

Um entsprechende Risiken vermeiden und Erfolgschancen für eine gezielte Marketingkommunikation eröffnen zu können, ist es deshalb unabdingbar, Veränderungen in den Lebensstilen und Kundenverhaltensmustern relevanter Zielgruppen im Lichte der unterschiedlichsten gesellschaftlichen Einflüsse systematisch zu analysieren. Ein wichtiger Trend bildet hierbei etwa die Veränderung des Freizeit- und Mediennutzungsverhaltens angesichts des technologischen Wandels und speziell der Entwicklungen im Bereich des Internets. Im Vergleich zu den klassischen Medien (TV, Zeitungen etc.) gewinnt so etwa die Nutzung interaktiver Medien (Internet etc.) immer mehr an Bedeutung (vgl. SevenOne Media 2005; Brown 2004; Müller-Kalthoff 2002). Zugleich bietet der skizzierte technologische Wandel die Fähigkeit und Bereitschaft zu einer Intensivierung eines Online-Dialogs bzw. einer Vernetzung mit den unterschiedlichsten Dialogpartnern (vgl. Röthlingshöfer 2006).

Im Zuge dieser Entwicklungen ist aus Marketingsicht besonders die Open-Source–Bewegung von Interesse. Als Pendant zur klassischen Softwareentwicklung gegründet (z. B. *Windows* vs. *Linux*), existieren diese Communitys heute als kollaborative Netzwerke in einer Vielzahl von Anwendungsgebieten, z. B. als Zusammenschlüsse von Kreativen, die Texte, Videos, Bilder oder Audio-Quellen generieren (z. B. *creativecommons.org*, *flickr.com* oder *jamendo.com*), als Bildungsnetzwerke (z. B. *MIT OpenCourseWare*) oder als kooperative Marketing-Communitys (z. B. *spreadfirefox.com* von *Mozilla*, *Vocalpoint* von *P&G* oder *conversegallery.com* von *Converse*).

Die Frage stellt sich nun, ob und ggf. inwieweit sich die OS-Bewegung zielorientiert nutzen lässt. Können also Unternehmungen oder auch nicht-kommerzielle Organisationen solche Prozesse des „Sich-aktiv-Einbringens" gezielt auslösen und/oder so steuern, dass sie zentrale Marketingziele besser verwirklichen können? Und an welche Voraussetzungen ist ein solches „Open-Source-Marketing" (OSM) gebunden?

Will man abschätzen, in welcher Weise gesellschaftliche Entwicklungen die Bereitschaft zu einem Engagement in OSM-Projekten beeinflussen und in welcher Weise parallel dazu Marketingtreibende ein solches Engagement ggf. auch gezielt beeinflussen können, kommt einer eingehenden Analyse der Motivstrukturen ein zentraler Stellenwert zu. Zwar liegen zur Frage, welche Motive zu einer Beteiligung an OS-Projekten führen können, bereits einzelne Beiträge vor, die erste Idee- und Denkanstöße für den Bereich des OSM geben können (vgl. Lakhani und Wolf 2002; Lerner und Tirole 2002; Gosh et al. 2002; Hars und Ou 2000), empirisch fundierte Erkenntnisse im speziellen Gebiet des OSM sind jedoch noch sehr dünn gesät. Zur Schließung dieser Forschungslücke soll im Folgenden ein Beitrag geleistet werden, indem zentrale Ergebnisse einer von uns durchgeführten Studie zu den Motiven einer Beteiligung an OSM-Projekten vorgestellt werden.

2 Entwicklung eines konzeptionellen Bezugsrahmens zur Erforschung der Motivstrukturen in OSM-Projekten

2.1 Zur Charakterisierung von OSM-Projekten

Vermehrt versuchen verschiedene Disziplinen, erfolgbringende OS-Wesenszüge und Eigenschaften (z. B. den freien Umgang mit Wissen oder das freie Nutzen kollektiv erstellter Entwicklungen) für ihre Zwecke zu übernehmen. Auch die Marketingwissenschaften beschäftigen sich seit jüngerer Zeit mit OS als Managementansatz. Ein anschauliches Fallbeispiel bietet die *Mozilla Foundation*. Das Non-Profit-Unternehmen befolgt zur Vermarktung seines Web-Browsers *Firefox* die Erfolgsregeln der OS-Bewegung, indem es u. a. eine Community für den konstruktiven Ideenaustausch schaffte, integrierte Mechanismen zur Auswahl möglicher Marketingaktivitäten etablierte und den größten Teil der strategischen Marketingplanung sowie operativen Umsetzung (wie z. B. Artwork, Werbekampagnen) in dem OS-Netzwerk entwickeln ließ (vgl. hierzu ausführlicher Mucha 2004; Lieb 2004). Die Ergebnisse des gemeinschaftlichen Marketings sind beträchtlich: Bis 10. Septemper 2007 zählten die Server von *Mozilla* mehr als 400 Millionen Downloads des Browsers.[1]

Aber nicht nur im Non-Profit-Bereich wird das Potenzial von OS geschätzt, auch privatwirtschaftliche Unternehmen entdecken vermehrt das OS-Konzept für ihre Marketingmaßnahmen. Pionier hier ist die zum Konzern *Nike* gehörende Marke *Converse*. Der Sportschuhhersteller initiierte den Filmwettbewerb „Leidenschaft für Schuhe". In den Kurzfilmen sollte der Verbraucher die Schuhe von *Converse* thematisieren. *Converse* lud auf diese Weise seine Verbraucher zur Gestaltung der Werbemaßnahmen ein.

In Folge der Kampagne verzeichnete nicht nur die *Converse* Firmenhomepage einen signifikanten Besucherzustrom (plus 400 000 Zugriffe monatlich), sondern auch die allgemeinen Umsatzzahlen erhöhten sich (plus 12 Prozent) merklich (vgl. Kiley 2005). Inzwischen sind eine ganze Reihe von Unternehmen wie *Red Bull*, *Daimler* oder *General Motors* dem Beispiel von *Converse* gefolgt und setzen auf vermehrte Integration ihrer Kunden in die Absatz- und Vermarktungsprozesse (vgl. Cherkoff 2005).

Vor diesem Hintergrund wird Open-Source-Marketing vorliegend als die unter Einbezug der Konsumenten gemeinschaftliche Entwicklung von Marketingideen und deren Umsetzungsmöglichkeiten auf Basis flexibler Nutzungsrechte definiert (vgl. Wiedmann und Langner 2006, 2007).

Das Open-Source-Marketing begründet somit ein neues Rollenverständnis des *marketers*. Die Funktion des *brand guardian*, der durch rigide Kontrolle jegliche Umdeutung der linearen Kommunikationsinhalte zu verhindern sucht, weicht der Rolle eines transparenzfördernden *brand host* (vgl. Cherkoff 2005; Brondmo 2004; McConnell

1 Siehe http://www.mozilla-europe.org/de/press/2007/09/10/935-400-millionen-downloads-von-mozilla-firefox. [10. Feb. 2008].

und Huba 2003). Er erleichtert es der global-dezentral organisierten Verbrauchergruppe, aktiv und freiwillig am Marketingprozess der Firmen teilnehmen zu können, indem gezielt u. a. konstituierende Barrieren wie Nutzungslizenzen liberalisiert, Derivate oder Weiterentwicklungen von bspw. Anzeigen und Logos gefördert sowie öffentliche Diskussionen der relevanten Marketingbestandteile in der Community angeregt werden (vgl. Wiedmann und Langner 2006, 2007).

Mit dieser proaktiven Art der Kundenansprache reagieren Unternehmen u. a. auf den geänderten Umgang der Konsumenten mit den Werbeinhalten. Verstärkt betreibt nämlich die heutige Verbrauchergeneration – meist durch das Internet als *enabler* – ihre eigene Art der Vermarktung (vgl. Oetting 2006). So stellen Privatleute auf Community-Portalen wie *youtube.com* oder *flickr.com* im Sinne eines „Open Sourcing Yourself" selbstproduzierte Inhalte – von persönlichen Fotos über kreativ-selbsterstellte Werbespots bis hin zu parodierten, persiflierten und rezyklierten Werbekampagnen, die Markenbotschaften der Firmen zweckentfremden – der Allgemeinheit zur Verfügung (vgl. Haaksman 2006; Parker 2006).

Zurückkommend auf die Beispiele von *Converse* und *Mozilla* wird eines deutlich: Konsumenten besitzen die Bereitschaft, sich im Rahmen von OSM zu engagieren. Die Ursachen für eine freiwillige Beteiligung können jedoch bislang lediglich vermutet werden. Im Folgenden stellt sich daher die Frage, welche Charakteristik ein OSM-Projekt besitzen muss, damit Konsumenten motiviert sind, sich im Rahmen der Vermarktung von Produkten und Dienstleistungen zu beteiligen.

2.2 Bezugsrahmen der konsumentenseitigen Motivation in OSM-Projekten

Die Forschungen zum Konstrukt der menschlichen Motivation sind sehr vielfältig und heterogen. Je nach gewählter Betrachtungsperspektive werden andere Aspekte der Motivation herausgegriffen und beleuchtet (vgl. hierzu ausführlicher Decy und Ryan 2000).

Zu den Triebfedern der individuellen Beteiligung an OSM akzentuieren wir nach Ramlall (2004) Motivation als Intermediär von Merkmalen der Person (Motive) sowie Merkmalen der Situation (Anreize) und rekurrieren zum weiteren Vorgehen auf zwei weitgehend feststehende Motivationsansätze:

Bedürfnisbasierte Motivation zielt auf die Identifikation der inneren Treiber und Faktoren, die individuelles Verhalten begründen, ab. Laut Definition sind Bedürfnisse zunächst physiologische oder psychologische Mangelzustände, die das Verhalten determinieren. Diese Bedürfnisse können vergleichsweise stark oder schwach sein und werden von Umweltfaktoren beeinflusst (vgl. Blackwell et al. 2001; Solomon et al. 2002).

Erwartungsbasierte Motivation bedeutet „[. . .] that people are motivated to behave in ways that produce desired combinations of expected outcomes" (Kretiner und

Kinicki 1998). Ein Individuum handelt daher in gewisser Weise gemäß der Erwartung, dass der eigenen Handlung ein bestimmter Ertrag folgen wird und entsprechend der Attraktivität dieses Ertrags für das Individuum (vgl. Vroom 1964).

Vor diesem theoretischen Hintergrund entwerfen wir ein konzeptionelles Modell zur Erklärung der Konsumentenbeteiligung in OSM-Projekten, das sich aus den drei Dimensionen der pragmatischen, sozialen und hedonistischen Motivation konstituiert (vgl. zur Modellkonzipierung ausführlicher Wiedmann, Langner und Pankalla 2007; Wiedmann, Langner und Hennigs 2007).

Die pragmatische Motivation umfasst diejenigen bedürfnis- und erwartungsorientierten Motive, die sich auf einen direkten Nutzen für den Konsumenten aus seiner Teilnahme an einem OSM-Projekt beziehen, wie z. B. der Erhalt einer spezifischen Entlohnung (*rewards*), die Verbesserung der individuellen Berufsaussichten (*reputation*), die Zusammenarbeit mit angesehenen Experten (*get in touch*) oder die Unterstützung bei anderen Projekten (*reciprocity*).

Die soziale Motivation bezieht sich auf diejenigen motivationalen Faktoren, die sich aus den interpersonalen Austauschbeziehungen innerhalb der Gemeinschaft ergeben, wie z. B. aus Identifikationsprozessen, gegenseitiger Anerkennung und Hilfe.

Die hedonistische Motivation integriert spezifische und nicht-spezifische emotionale Treiber für die Beteiligung an einem marketingorientierten OS-Netzwerk, wie z. B. die Begeisterung für oder emotionale Anziehungskraft einer Marke oder positive Erfahrungen aus dem Arbeiten innerhalb einer Gruppe (*fun & flow*).

Aufgabe der empirischen Untersuchung ist es nun, zu bestimmen, inwieweit die drei beschriebenen Grunddimensionen bestätigt, welche einzelnen Faktoren allgemein identifiziert werden, und ob eine feinere Dimensions- und Konstruktzerlegung zur Beschreibung von OSM-Motivation notwendig sein wird.

3 Empirische Analyse der Motivstrukturen in OSM-Projekten

3.1 Art und Anlage der empirischen Untersuchung

Zur Untersuchung der konsumentenseitigen Motivation in OSM-Projekten wurde ein professioneller deutscher Community-Provider gewählt, der auf viel Erfahrung bei der Durchführung von OSM-Projekten verweisen kann. Um die meisten der zahlreichen OSM-Facetten abzudecken, lag der spezielle Fokus der Untersuchung auf drei unterschiedlichen OSM-Projekten großer und bekannter B2C-Unternehmen aus

Deutschland. Im Einzelnen wurden die Teilnehmer aus Projekten des Discount-Handytarifanbieters *Simyo*, des Bonusprogramms *payback* und des führenden Anbieters personalisierter Bekleidung *Spreadshirt* rekrutiert.

Als Instrument der Erhebung kam eine Online-Befragung zum Einsatz. Alle in der Studie verwendeten Items basierten auf bestehenden und bereits getesteten Skalen aus vorangegangenen Studien und Untersuchungen zur Motivation in OSN (vgl. hierzu ausführlicher Wiedmann, Langner und Pankalla 2007). Die Formulierung der Skalen wurde den Charakteristika von OSM-Projekten angepasst. Um OSM-spezifische Motivationsaspekte abfragen zu können (wie beispielsweise Markenbegeisterung), wurden zudem weitere Items auf Basis von qualitativen Interviews mit Projektteilnehmern und Experten generiert. Bei allen Items kam eine 5-Punkte-Likert-Skala zum Einsatz. Der Fragebogen wurde mittels explorativer Interviews und zwei Pre-Tests (online und offline) auf Validität überprüft.

Nur Teilnehmern der drei ausgewählten Communitys war es erlaubt, an der Befragung teilzunehmen. Die Einladung zur Umfrage erfolgte über einen Link auf der Homepage der jeweiligen Community und eine personalisierte E-Mail. Alle Teilnehmer mussten sich mit ihrem Login und ihrem Passwort authentifizieren. So wurde sichergestellt, dass nur tatsächliche Mitglieder des jeweiligen OSM-Projekts an der Befragung teilnahmen. Von 483 zur Befragung eingeladenen Teilnehmern nahmen 246 Personen teil, was einem Rücklauf von 51 Prozent entspricht.

Mittels (explorativer) Faktorenanalyse wurden die erhobenen Daten ausgewertet. Die gewählte Faktorlösung offenbart dabei eine Struktur von acht inhaltlich zu extrahierenden Faktoren, die insgesamt 69 Prozent der ursprünglichen Varianz erklären. Das Kaiser-Meyer-Olkin-Maß beträgt 0,89 und das Cronbach's Alpha der Faktoren weist Werte von 0,69 bis 0,91 auf, was auf eine hinreichend gute Reliabilität und eine entsprechende Verallgemeinerbarkeit der Ergebnisse in Bezug auf die Motivation von OSM-Teilnehmern schließen lässt. Alle Faktoren mit niedrigeren Cronbach's Alpha Werten (< 0,6) wurden von weiteren Analysen ausgeschlossen (vgl. für eine ausführliche Darstellung der mathematisch-statistischen Auswertung Wiedmann, Langner und Pankalla 2007).

3.2 Zentrale Ergebnisse der Untersuchung und Ansatzpunkte für das Marketing

Wichtige Aussage unserer Analyse ist, dass Motivation in OSM-Projekten auf einer komplexen und vielschichtigen Zusammensetzung von eindeutig unterscheidbaren Faktoren beruht und zu einer differenzierteren Auffächerung als der drei konstruierten Motivationsdimensionen tendiert. Die Initiierung eines OSM-Projekts erfordert somit eine gewisse Vielfalt unterschiedlicher Stimuli und ein Gespür dafür, diese in der richtigen Mischung und Intensität zu verwenden.

Auffällig ist auch, dass nahezu kein pragmatisch orientiertes Motiv empirisch gefunden werden konnte. In diesem Kontext in der OS-Literatur häufig genannte moti-

vationale Faktoren wie Reziprozität oder materielle bzw. finanzielle Entlohnung (vgl. u. a. Watson 2005; Lakhani und von Hippel 2002) sind als Triebfedern im OSM–Bereich nicht zu erkennen. OSM-Teilnehmer streben demzufolge mit ihrem Wirken keine direkte Vergütung an und spekulieren für ihren geleisteten Arbeitsaufwand nicht taktisch-opportunistisch auf Gegenleistungen im Sinne eines *quid pro quo* durch die anderen OSM-Teilnehmer. Im Folgenden werden alle identifizierten Faktoren vorgestellt und hinsichtlich möglicher Gestaltungsansätze diskutiert.

Faktoren im Kontext einer stärker pragmatisch orientierten Motivation

Faktor 1: Learning and Stimulation Unsere empirische Erhebung ergab, dass OSM-Projekte mit komplexer Problemstruktur, aktiver *peer review* und daraus ermöglichtem *learning by doing* (d. h. bereits theoretisch erworbenes Wissen findet Anwendung in praktischem Rahmen) bzw. *explorative learning* (d. h. stimulierende, neue Lösungsansätze für bereits existierende Probleme werden gefunden) motivationale Treiber sind. So scheinen OSM-Teilnehmer durch ihr Mitwirken an der Entwicklung von Marketingideen und deren Umsetzung, Fähigkeiten und Qualitäten vermittelt zu bekommen, die für sie im Beruf und ihre weitere Karriere relevant sein können. Dieses Ergebnis zeigt Ähnlichkeiten zu Untersuchungen im OS-Bereich. Hier wurde herausgefunden, dass das Erstellen von Programmcode von den Teilnehmern häufig als sehr fordernd, intellektuell stimulierend und für andere Zwecke nützlich beschrieben wird (vgl. Lakhani und von Hippel 2002; Lakhani und Wolf 2002; Lerner und Tirole 2002). Aus Marketingsicht wird man diesem Teilnahmemotiv in der OSM–Praxis bisher noch wenig gerecht. Damit aber lernbegierige Personentypen, wie z. B. der legendär gewordene iPod-Werbespotentwickler George Masters, dessen originäres Ziel nach eigenen Aussagen darin bestand, seine Fähigkeiten in Grafikdesign und Computeranimation zu verbessern, einen Anreiz zum Mitmachen bei OSM bekommen würden, mag ein denkbarer Ansatzpunkt im Implementieren einer (Online-)Plattform sein, auf der OSM-Teilnehmer mit ausgewiesenen Experten aus speziellen Bereichen des Marketing oder anderer Disziplinen Wissen und Know-how austauschen können.

Faktor 2: Get-in-Touch Vor dem Hintergrund unserer Ergebnisse wird deutlich, dass ein Engagement bei OSM-Projekten auch durch das Entstehen neuer Kontakte und das Vergrößern seines sozialen Netzwerks motivational begründet werden kann. Speziell das Kennenlernen von und die Diskussion auf Augenhöhe mit hochrangigen Funktionsträgern bzw. ausgewiesenen (Marketing-)Experten stellen erhobene Teilnahmemotive dar. Die Beziehungen zu diesen Akteuren werden seitens der OSM-Teilnehmer dabei häufig zur eigenen, mittelbaren und instrumentellen Bedürfnisbefriedigung (bspw. ein Bewerber beruft sich gegenüber dem Personalchef auf gemeinsame Bekannte) umfunktioniert, weshalb die

Teilnahme an einem OSM-Projekt die Wesenszüge eines (langfristigen) Investments aufweisen kann. Ein gutes Vorbild zur Erhöhung dieses Motivs bietet die Firma *Frosta*. Diese initiierten ein Weblog, das nicht nur für die Mitarbeiter-interne Kommunikation bestimmt ist, sondern das auch und gerade für den direkten Dialog der gesamten *Frosta* Belegschaft mit den Verbrauchern verwendet wird.

Faktoren im Kontext einer stärker sozial orientierten Motivation

Faktor 3: Peer Recognition Das Verlangen nach Anerkennung der eigenen Persönlichkeit und des eigenen Schaffens ist ein mit dem menschlichen Selbstbewusstsein zwingend verknüpfter Antrieb (vgl. Popitz 1987, S. 633). Erst die von Mitmenschen erlangte Aufmerksamkeit formiert letztendlich das persönliche Ansehen und bewirkt damit eine nachhaltige Ausprägung des individuellen Selbstwertgefühls (vgl. Franck 1998, S. 7). Unsere Studie zeigt, dass das OSM dabei grundsätzlich keinen Ausnahmefall darstellt. Persönliche Wertschätzung, artikuliert durch anerkannte Autoritäten oder einen „Kreis von Gleichgesinnten" (*peergroup*), und die mögliche Selbstdarstellung innerhalb der OSM-Community erhielten besondere motivationale Beachtung. Möglicher Gestaltungsansatz für das Marketing könnte hierbei die Erstellung einer (Online-)„Hall of Fame" sein, in der beispielsweise sämtliche konsumentengenerierten Werbespots archiviert und für jeden anderen zugänglich gemacht werden. *Best practice* ist in diesem Zusammenhang die Marke *Converse*. Diese stellen alle zu ihrem Filmwettbewerb eingesendeten Werbeclips (mit Namen und E-Mail-Adresse des Erschaffers versehen) in der *conversegallery.com* online und sorgen somit für den nötigen *geek fame* bei ihren Teilnehmern.

Faktor 4: Altruismus Die vorliegende Untersuchung konnte eine wertbasierte, altruistische Haltung der Teilnehmer als eine Motivationsvariable nachweisen. Anderen OSM-Mitgliedern mit dem eigenen Marketingfachwissen und spezifischen Ideen helfen zu können, ihre Unternehmen oder Projekte beispielsweise durch eine neue Marketingtechnik zu verbessern, kann zur individuellen Bedürfnisbefriedigung beitragen, insbesondere wenn dies durch das Internet zu geringen Transaktionskosten geschieht. In einer beliebten Domäne zu arbeiten und damit altruistisch agieren zu können, ist auch in OS-Projekten ein häufig zu findendes Teilnahmemotiv (vgl. u. a. Bergquiest und Ljungberg 2001; Raymond 1999).

Faktor 5: Community Identification Je stärker sich ein Individuum mit einem OS-Projekt identifizieren kann, desto höher ist auch die Wahrscheinlichkeit der Teilnahme. Diese Feststellung beruht auf der Annahme eines starken gemeinschaftlichen Glaubens- und Identitätsgefühls innerhalb von OS-Communitys (vgl. Raymond 1999; Levy 1994). Viele OSS-Programmierer beispielsweise schätzen nicht nur die Idee von freier Software, sie leben sie regelrecht. Diese „Art" von

Programmierer zieht ein spezifisches OS-Projekt nur in Erwägung, wenn sie gleichgesinnte Community-Mitglieder erwartet und findet (vgl. Weber 2004). Ähnliches, wenn auch gleich weniger ideologisch geprägt, konnten wir empirisch in OSM-Communitys feststellen. Auch hier identifizieren sich die Nutzer mit der Gemeinschaft und wollen „Teil der außergewöhnlichen Community" sein, dabei liegt der Fokus aber weniger auf dem Zielhorizont, sondern vielmehr auf der „besonderen Form der Zusammenarbeit". Mit Blick auf dieses Motiv sollte die erste Marketingmaßnahme in der Schaffung eines Sozialisationspunktes (Community) liegen, sodass die Konsumenten untereinander in einen konstruktiven Ideenaustausch gelangen können. Beispiele hierfür sind Brand-Communitys wie die von *Nintendo*, *Jägermeister* oder *Harley Davidson*.

Faktoren im Kontext einer stärker hedonistisch orientierten Motivation

Faktor 6: Joint Enemy/Consumer Empowerment Die Teilnahme an einem OS-Projekt ist bei vielen Community-Mitgliedern verknüpft mit dem Kampf für ein gemeinsames Ziel bzw. gegen einen gemeinsamen „Gegner" (vgl. Lerner und Tirole 2002; Gosh et al. 2002). Die dahinterstehende Motivation beruht auf ideellen Ansichten bezüglich Geschäftspraktiken und ästhetischen Aspekten sowie dem Glauben an Freiheit und Unabhängigkeit (z. B. bei der Erweiterung von Software) (vgl. Weber 2004). Wie unsere Untersuchung zeigt, haben Teilnehmer an OSM-Projekten eine ähnliche Einstellung. Sie denken, dass „Marketing [. . .] viel besser [ist], wenn die Konsumenten aktiv bei der Entwicklung beteiligt werden" bzw. sind der Überzeugung, dass im Idealfall „[. . .] Marketingmaßnahmen durch die Konsumenten entworfen werden" sollten. Diese Aussagen beweisen, dass nicht allein herausfordernde Aufgaben und eine funktionierende Gemeinschaft als Motivation für OSM-Teilnehmer nötig sind, sondern auch eine grundsätzliche Einstellung zur konsumentenseitigen Integration in Marketingprozesse muss im Rahmen des Projekts kommuniziert werden. Anders ausgedrückt, brauchen OSM-Mitglieder das Gefühl, dass ihre Beteiligung die OSM-Ergebnisse wesentlich mitbestimmt und ihre Ideen ernst genommen werden.

Faktor 7: Markenbegeisterung Für OSN-Mitglieder ist nicht nur relevant, was sie tun, sondern vielmehr für wen. Hierbei zeigt sich die starke motivationale Wirkung der Markenbegeisterung. Einige Marken (wie *Firefox* oder *Apple*) haben die emotionale Kraft, ihre Kunden auf eine Weise zu aktivieren, dass diese sich freiwillig in unternehmensrelevanten Communitys wie Internetforen, Chats (z. B. IRCs) oder sogar markenbezogenen Fanklubs engagieren (vgl. Roberts 2005; McConnell und Huba 2003). Gleiches gilt für die hier betrachteten OSM-Projekte. Markenbegeisterung stellt folglich einen starken motivationalen Treiber

für die individuelle Bereitschaft einer Beteiligung an einem OSM-Projekt dar. Gestaltungsansatz zur Erhöhung dieser Teilnahmemotivation könnte u. a. sein, systematisch diesem speziellen Kundenkreis Privilegien wie beispielsweise exklusive Produkttests vor dem eigentlichen *going public* einzuräumen, womit die Bindung zum Unternehmen erhöht und Multiplikatoreneffekte generiert werden könnten.

Faktor 8: Emotional Appeal Sein kreatives Talent und seine künstlerische Freude durch OSM-Projekte verwirklichen und ausleben zu können, ist in unserer durchgeführten Studie ein elementarer Motivationsfaktor. Damit werden ein weiteres Mal Parallelen zur OS-Software sichtbar. Denn viele OS-Entwickler streben ebenfalls elegante, schnelle oder „einfach geniale" Lösungen an (vgl. Torvalds und Diamond 2001). Nicht nur die rein zweckmäßige Applikationsproduktion, sondern eine gleichzeitig höchstästhetisch anmutende Leistungserstellung gewinnt innerhalb der OS-Gemeinde immer mehr an Bedeutung. Das Verfassen virtuoser Programme stilisiert mitunter zum Mittel des Selbstausdrucks und der Positionierung innerhalb der Community (vgl. Weber 2004; Lakhani und von Hippel 2002; Amabile 1998), weshalb es wenig verwundert, dass einige OS-Teilnehmer das Schaffen von Programmcode als *intellectually stimulating* und mit dem Kreieren von Poesie oder Musik vergleichen (vgl. Lakhani und von Hippel 2002). Zur Steigerung dieses Motivs ergibt sich aus Marketingsicht ein ähnlicher Ansatz wie beim Faktor *peer recognition*, allerdings hier mehr mit der Akzentuierung auf der Ausgestaltung bzw. Aufmachung eines Kreativwettbewerbs (z. B. Erstellung von „Consumer Generated Media" zu realen Werbekampagnen) und weniger auf der Möglichkeit, sein „Machwerk" danach in der Community präsentieren zu können.

4 Fazit und Ausblick

Unser Forschungsvorhaben war vom Bedarf nach mehr Klarheit im Hinblick auf eine Konzeptualisierung und Messung der OSM-Motivation getrieben. Zentrale Ergebnisse der durchgeführten Studie lagen zum einen in der Erkenntnis, dass Motivation im OSM-Bereich ein vielschichtiges, Multi-Faktoren-basierendes Konstrukt ist und nicht allein durch einen einzelnen motivationalen Treiber varianzlos erklärt wird und zum anderen darin, dass primär sozial-intrinsische Motive verantwortlich für eine individuelle Teilnahmeentscheidung zu sein scheinen. Diesem Umstand kommt gerade mit Blick auf die Projektausgestaltung und die Erreichung relevanter Konsumentensegmente sowie Zielgruppen besondere Bedeutung zu.

Da viele der erhobenen Motivationsfaktoren zur OSM-Teilnahme den OSN-Treibern im Allgemeinen und den Motiven der OSS-Entwickler im Besonderen ähnlich sind, können marketingorientierte OS-Netzwerke viele der Erfolgsgrößen anderer

OS-Projekte übernehmen (wie z. B. effiziente Community-Austauschprozesse, Verhaltensregeln oder Aspekte der Usability), die soziale Motive wie Community-Identifikation, *peer recognition* oder hedonistische Aspekte wie *emotional appeal* aktivieren. Spezifische OSM-Faktoren wie Markenbegeisterung bedürfen einer differenzierteren Herangehensweise durch das Marketingmanagement. Allerdings muss gerade die Wichtigkeit von Markenbegeisterung und deren Abhängigkeiten detaillierter analysiert werden, da neue Marken oder Unternehmen vielleicht nicht ausreichend Zugkraft besitzen, um genügend Community-Mitglieder anzuziehen.

Zweifelsohne sind die Ergebnisse unserer Untersuchung nur ein erster Schritt und sollten in mehrfacher Hinsicht weiter entwickelt werden. Dies schließt ein, dass weiterführende Forschungen den Weg zu einer konfirmatorischen Überprüfung ebnen müssen, damit detaillierte, verifizierbare Aussagen über Ursache-Wirkungs-Beziehungen zu treffen sind und letztendlich der Entwurf einer verwendbaren Skala zur Erfassung der OSM-Motivation zu erreichen ist. In diesem Kontext sind die Motivationsdimensionen noch einmal in detaillierterer Form zu operationalisieren.

Bedarf zur weiteren Forschung liefert auch die Frage, ob die OS-Bewegung im speziellen Bereich des Marketing bereits eine relevante Segmentstärke erreicht hat bzw. künftig erreichen wird und inwieweit Motive und Faktoren sich ändern bzw. eine andere Gewichtung erfahren, wenn die OSM-Bewegung mehr Anhänger gewinnt. Auch die Auswirkungen situativer Aspekte wie gesellschaftsorientierter Phänomene (z. B. Trend zur Erlebnisgesellschaft) oder die Auswirkungen neuer technologischer Entwicklungen und ihrer Implikationen auf die Zusammensetzung der Dimensionen und Faktoren sind in der Zukunft zu untersuchende Aspekte. Ungeachtet dieser Einschränkungen und dem notwendigen Forschungsbedarf ist es ein wesentlicher Beitrag der vorliegenden Untersuchung gewesen, eine erste Analyse und Taxonomie der Motivation in OSM-Projekten zu geben.

Literatur

Amabile, T. (1998), 'How to kill creativity?', *Harvard Business Review* **76**(5), S. 76–87.

Bergquiest, M. und Ljungberg, J. (2001), 'The power of gifts: organizing social relationships in Open Source communities', *European Journal of Information Systems* **11**(4), S. 305–320.

Blackwell, R. D., Miniard, P. W. und Engel, J. F. (2001), *Consumer behavior*, 9. Aufl., Harcourt College Publishers, Fort Worth.

Brondmo, H. (2004), 'Open-Source-Marketing', http://www.clickz.com/experts/brand/sense/article.php/3397411 [10. Feb. 2008].

Brown, M. (2004), EIAA-Studie zur Mediennutzung 2003, White paper, European Interactive Advertising Association. http://eiaa.net/Ftp/casestudiesppt/White%20paper%20-%20German%2007_07.pdf [10. Feb. 2008].

Cherkoff, J. (2005), 'You, too, can become a global broadcaster', *The Observer* **31. Jul. 2005**.

Decy, E. und Ryan, R. (2000), 'Die Selbstbestimmung der Motivation und ihre Bedeutung für die Pädagogik', *Zeitschrift für Pädagogik* **45**(2), S. 223–238.

Franck, G. (1998), *Ökonomie der Aufmerksamkeit*, Hanser, München.

Gosh, R., Krieger, B., Glott, R. und Robles, G. (2002), FLOSS – Free/Libre and Open Source Software, Survey and Study commissioned by the EU, International Institute of Infonomics, University of Maastricht und Berlecon Research GmbH. http://www.infonomics.nl/FLOSS/report/ [04. Feb. 2008].

Haaksman, D. (2006), 'Versende deine Jugend', *Frankfurter Allgemeine Zeitung* **19. Mai 2006**. http://www.faz.net/s/Rub4C34FD0B1A7E46B88B0653D6358499FF/Doc~EC1B5703428E0470F95C9001B0B04D59A~ATpl~Ecommon~Scontent.html [10. Feb. 2008].

Hars, A. und Ou, S. (2000), 'Working for free? Motivations for participating in Open-Source Projects', *International Journal of Electronic Commerce* **6**(3), S. 25–39.

Kiley, D. (2005), 'Advertising of, by, and for the people', *BusinessWeek* **25. Jul. 2005**. http://www.businessweek.com/magazine/content/05_30/b3944097.htm [10. Feb. 2008].

Kretiner, R. und Kinicki, A. (1998), *Organizational Behavior*, 4. Aufl., Irwin, Chicago.

Lakhani, K. und Wolf, B. (2002), The Boston Consulting Group/OSTG: Hacker Survey, techn. Bericht, The Boston Consulting Group.

Lakhani, K. und von Hippel, E. (2002), 'How open source software works: ?free? user-to-user assistance', *Research Policy* **31**(2), S. 1–21.

Lerner, J. und Tirole, J. (2002), 'Some simple economics of open source', *Journal of Industrial Economics* **50**(4), S. 197–234.

Levy, S. (1994), *Hackers. Heroes of the computer revolution*, Delta, New York.

Lieb, R. (2004), 'Crazy Like a Firefox', http://www.clickz.com/showPage.html?page=3434811 [10. Feb. 2008].

McConnell, B. und Huba, J. (2003), *Creating Customer Evangelists. Discussion Guide*, Dearborn Trade Publishing, New York.

Mucha, T. (2004), 'Firefox: Marketing's Borg - The new browser taps the power of the collective', Online Dokument.

Müller-Kalthoff, B. (2002), Cross Media als integrierte Management-Aufgabe, *in* B. Müller-Kalthoff (Hrsg.), 'Cross-Media-Management: Content-Strategie erfolgreich umsetzen', Springer, Berlin, S. 19–40.

Oetting, M. (2006), Wie Web 2.0 das Marketing revolutioniert, *in* T. Schwarz und G. Braun (Hrsg.), 'Leitfaden Integrierte Kommunikation', Absolit, Waghäusel, S. 173–200. http://www.connectedmarketing.de/downloads/oetting_wie-web20-das-marketing-revolutioniert.pdf [10. Feb. 2008].

Parker, P. (2006), 'You and Your Users, Marketing Together', http://www.clickz.com/showPage.html?page=3600706 [10. Feb. 2008].

Popitz, H. (1987), 'Der Wandel der Sozialen Subjektivität', *Kölner Zeitschrift für Soziologie und Sozialpsychologie* **39**(4), S. 633–647.

Ramlall, S. (2004), 'A review of employee motivation theories and their implications for employee retention within organizations', *The Journal of American Academy of Business* **5**(1/2), S. 52–63.

Raymond, E. S. (1999), *The Cathedral and the Bazaar: Musings on Linux and Open Source from an Accidental Revolutionary*, O'Reilly & Associates, Cambridge. http://www.catb.org/~esr/writings/cathedral-bazaar/ [10. Feb. 2008].

Roberts, K. (2005), *Lovemarks – the future beyond brands*, powerHouse Books, New York.

Rose, F. (2004), 'The lost boys', *Wired* **12**(8). http://www.wired.com/wired/archive/12.08/lostboys.html [10. Feb. 2008].

Röthlingshöfer, B. (2006), *Marketeasing. Werbung total anders*, Verlag Erich Schmidt, Berlin.

SevenOne Media (2005), 'TimeBudget 12', http://appz.sevenonemedia.de/download/publikationen/TimeBudget12.pdf [04. Feb. 2008].

Solomon, M., Bamossy, G. und Askegaard, S. (2002), *Consumer behaviour: A European perspective*, 2. Aufl., Financial Times Prentice Hall, Harlow.

Spitzer, R. und Swidler, M. (2003), 'Using a Marketing Approach to Improve internal Communications', *Employment Relations Today* **30**(1), S. 69–82.

Torvalds, L. und Diamond, D. (2001), *Just for Fun. Wie ein Freak die Computerwelt revolutionierte*, Hanser, München.

Vroom, V. H. (1964), *Work and Motivation*, Wiley, New York.

Watson, A. (2005), Reputation in Open Source Software, Working paper, College of Business Administration, Northeastern University.

Weber, S. (2004), *The success of open source*, B&T, Cambridge.

Wiedmann, K.-P. und Langner, S. (2006), Open Source Marketing – ein schlafender Riese erwacht, *in* B. Lutterbeck, M. Bärwolff und R. A. Gehring (Hrsg.), 'Open Source Jahrbuch 2006', Lehmanns Media, Berlin, S. 139–150.

Wiedmann, K.-P. und Langner, S. (2007), Open Source als Herausforderung an das Marketing-Management, *in* H. Bauer, D. Große-Leege und J. Rösger (Hrsg.), 'Interaktives Marketing im Web 2.0', Vahlen Verlag, Mannheim, S. 127–150.

Wiedmann, K.-P., Langner, S. und Hennigs, N. (2007), Open Source Marketing: Ergebnisse einer empirischen Analyse der Motive des freiwilligen Konsumentenengagements in gemeinschaftlichen Marketing-Projekten, *in* B. Hass und G. W. T. Kilian (Hrsg.), 'Web 2.0. Neue Perspektiven für Marketing und Medien', Springer, Berlin, S. 231–248.

Wiedmann, K.-P., Langner, S. und Pankalla, L. (2007), *Open Source Marketing: Die gemeinschaftliche Entwicklung von Marketingstrategien und -taktiken mit dem Konsumenten – eine empirische Analyse zur Erklärung der verbraucherseitigen Motivation*, Universität Hannover – Institut für Marketing & Management, Hannover.

Der Fall Microsoft – Offengelegte Schnittstellen und offengebliebene Fragen

LEONIE BOCK

Der Beitrag befasst sich mit dem Verhältnis zwischen Kartell- und Urheberrecht. Gegenstand der Betrachtung ist die Zwangslizenz nach Art. 82 EG-Vertrag (EG) gegen den Inhaber geistigen Eigentums, dessen Recht wesentlich ist für die Teilnahme an einem nachgelagerten Markt. Anlässlich des Urteils des Europäischen Gerichts erster Instanz (EuG) vom 17. 9. 2007 im Verfahren gegen Microsoft werden die von der europäischen Rechtsprechung entwickelten Tatbestandsvoraussetzungen kritisch dargestellt. Besonderes Augenmerk wird hierbei auf das Kriterium der *Verhinderung eines neuen Produkts* gerichtet. Dieses ist vor dem Hintergrund des Schutzzwecks von Art. 82 notwendigerweise so auszulegen, dass darauf aufbauend die Verhinderung des Wettbewerbs um das Produkt als Voraussetzung für die Zwangslizenz gefordert wird.

Schlüsselwörter: Microsoft · Monopol · Europäisches Recht

1 Einleitung

„Was hat es gebracht?" – diese Frage wird man nach dem Abschluss des langjährigen Verfahrens gegen Microsoft durchaus stellen können, wenn auch Microsoft selbst ihr einen eher rhetorischen Sinn beimessen dürfte. Aus Windows-Betriebssystemen wird auch nach dem Urteil des EuG vom 17. 9. 2007[1] keine Open-Source-Software werden. Ergebnis der erzwungenen Offenlegung von Schnittstelleninformationen wird jedoch eine Software sein, die sich dem Interesse von Verbrauchern und Wettbewerbern

1 Entscheidung des EuG in der Rechtssache T-201/04, 2007/C269/80 (operative part of the judgment); in englischer Sprache abrufbar unter http://curia.europa.eu/en/content/juris/t2.htm [06. Feb. 2008]; im Folgenden zitiert als Microsoft-Entscheidung EuG.

öffnet, indem sie eine erweiterte Kompatibilität durch die mögliche Verknüpfung mit Konkurrenzprodukten gewährleistet.

Ein weiteres Ereignis mag Anlass geben, den Fall Microsoft gerade im Zusammenhang mit Open Source zu betrachten: Der *Netscape Navigator*, einst von Microsoft als ernsthafte Konkurrenz wahrgenommen und, abgesehen von einem anfänglichen Marktanteil von bis zu 85 Prozent, auch durch seine Rolle bei der Eröffnung des Verfahrens gegen Microsoft in den USA bekannt geworden, ist am Ende.[2] Schon ab dem Jahr 1999 hatte Microsoft durch seinen Internet Explorer den Browsermarkt wieder unter Kontrolle; nun stellt der derzeitige Inhaber des *Netscape Navigator*, *AOL*, zum 1. 2. 2008 die Veröffentlichung der Sicherheitsupdates ein.

Der vorliegende Beitrag behandelt die Frage, welche Erkenntnisse das Verfahren gegen Microsoft für die Beurteilung zukünftiger, ähnlich gelagerter Fälle aus der Sicht des Kartell- und Urheberrechts gebracht hat. Gibt es nunmehr klare Tatbestandsvoraussetzungen für die Erteilung einer Zwangslizenz nach Art. 82 EG gegen den Schutzrechtsinhaber, der in der besonderen Lage ist, ein für die Marktteilnahme wesentliches Recht innezuhaben?

2 Microsoft und Open Source

Die Möglichkeit, Softwareprodukte herzustellen, die auch mit denen anderer Hersteller kompatibel sind, vereinfacht den Anbietern den Marktzugang und erweitert die Auswahl für die Nutzer dieser Produkte. Dies wiederum führt zu einer erweiterten Produktselektion und letztlich auch zu mehr Wettbewerb, da die konkurrierenden Softwarehersteller sich der Nachfrage der Verbraucher anpassen müssen, um auf dem Markt zu bestehen. Auch wenn dieser Mechanismus lediglich auf die Wahl zwischen den fertigen Produkten verschiedener Hersteller bezogen ist und nicht schon auf die Gestaltung der Produkte im Entwicklungsprozess selbst ausgedehnt wird – Freiheit in Angebot und Nachfrage durch die Offenlegung von Arbeitsergebnissen ist zumindest doch ganz im Sinne der Open-Source-Philosophie (Freyermuth 2007, S. 26, 30, 35 ff.). Denn wenn Freiheit des Wettbewerbs mit der Freiheit des Wettbewerbs um Software gleichzusetzen ist, Software aber wiederum ein Medium zur Distribution von Information bildet, ist das Kernthema der Open-Source-Bewegung getroffen.

Der Fall Microsoft entspricht nicht den üblicherweise beim Zusammentreffen von Open Source und Kartellrecht relevanten Konstellationen, bei denen es etwa um die Bewertung der Gebührenfreiheit beziehungsweise der Rücklizenzierung nach Art. 81 EG geht. Auch wird im Folgenden nicht näher auf das Verhältnis von Open Source und dem Urheberrecht an Software – ob man nun erstere als Ausformung des Urheberpersönlichkeitsrechts oder als Gegensatz zum Urheberrecht im Ganzen sieht – eingegangen werden (siehe dazu jeweils Jaeger und Metzger 2006, S. 207 ff., 82 ff.). Die Entscheidung des EuG gibt vielmehr Gelegenheit, die Abgrenzung von Urheber-

2 Siehe Frankfurter Allgemeine Zeitung vom 31.12.2007, S. 17.

und Kartellrecht genauer zu betrachten: Wo die Verweigerung einer Lizenz aufgrund der besonderen Marktsituation als Missbrauch im Sinne von Art. 82 EG einzustufen ist, stellt sich die Frage nach dem Ob und Wie eines kartellrechtlichen Eingriffs in Immaterialgüterrechte.

Open-Source-Software, deren kommerzieller Nutzen für den Urheber nicht in dessen ausschließlichem Recht zur Lizenzerteilung liegt (Jaeger und Metzger 2006, S. 83), nimmt durch die uneingeschränkte und garantierte Lizenzierung und allseitige Weiterentwicklungsmöglichkeit dem Monopol eine wichtige Voraussetzung und dient damit indirekt den Zielen des Kartellrechts. Eine Marktverdrängung durch technische Software-Modifikationen oder das mutwillige Blockieren von Vertriebskanälen[3] wären im Fall offener Quellcodes so nicht möglich gewesen.

In der Situation eines Urhebers so genannter proprietärer Software, der sein ausschließliches Recht gerade nicht durch Offenlegung, sondern durch die kommerzielle Weitergabe limitierter Information ausübt, ist das Kartellrecht, ausgehend von einem Markt und anknüpfend an wirtschaftliche Macht, dagegen grundsätzlich anwendbar. Maßgeblich ist die wettbewerbsverfälschende Wirkung, die der Urheber in der besonderen Konstellation des Monopols durch die Ausgestaltung der Lizenzierung hervorruft und die in der eingeschränkten Auswahl beim Verbraucher und durch erschwerten Marktzutritt bei den Wettbewerbern ihre Auswirkungen zeigt. Auch wenn beispielsweise die rein technische Koppelung mehrerer Softwareprodukte eines Herstellers[4] auf die entsprechende Beeinflussung der Abnehmer (und nicht unmittelbar der Wettbewerber) abzielt und deshalb als Marktmachtmissbrauch schwer zu erfassen ist – Ziel des Kartellrechts ist nicht vordergründig die optimale Versorgung des Verbrauchers mit Produkten, sondern die Entwicklung der Produktauswahl im unverfälschten Wettbewerb, d. h. eben auch der Schutz der konkurrierenden Anbieter in ihrer unternehmerischen Freiheit.[5]

3 Der Fall Microsoft im Einzelnen

Im Fall Microsoft stellte die Kommission zunächst fest, der Missbrauch einer marktbeherrschenden Stellung nach Art. 82 EG liege darin, dass Microsoft die Kompatibilität seiner Software mit der Software konkurrierender Unternehmen durch Geheimhaltung technischer Daten (Schnittstellen) bewusst eingeschränkt habe. Streitgegenständlich waren so genannte Schnittstelleninformationen, d. h. technische Daten, welche die Kommunikation zwischen einzelnen PCs oder die gemeinsame Nutzung von Dateien, Druckern oder auch Internetverbindungen und damit die Interoperabilität von Mi-

3 So von Microsoft als Reaktion auf die Entwicklung von Middleware vorgenommen (Ishii und Lutterbeck 2002, S. 140 ff.).

4 Etwa die Verknüpfung des Internet Explorers mit Windows (siehe dazu Ishii und Lutterbeck 2002, S. 145 ff.).

5 Siehe hierzu grundlegend Hirsch et al. (2007, Säcker, Einleitung, Rn. 4, 12); Mestmäcker und Schweitzer (2004, Möschel, Art. 82, Rn. 5); Bunte und Langen (2006, Dirksen, Art. 82, Rn. 1 ff.).

crosoft-PC-Betriebssystemen mit Server-Betriebssystemen anderer, konkurrierender Hersteller möglich machen (Schricker 2006, Loewenheim, § 69a, Rn. 13). Die Schnittstellen bildeten aus der Sicht der Kommission eine wesentliche Einrichtung im Sinne der Essential-Facilities-Doktrin und der Zugang zu ihnen die unabdingbare Voraussetzung für die Wettbewerber von Microsoft, um auf dem Markt für kompatible Systeme zu bestehen.

3.1 Gang des Verfahrens

Zunächst wurde der Fall Microsoft in den USA verhandelt. Nach der Klage der Regierung sowie mehrerer Bundesstaaten 1998, in der Microsoft der Missbrauch seiner Monopolstellung vorgeworfen wurde, wurde Microsoft zur Aufspaltung des Unternehmens verurteilt. In Gang gesetzt wurde das Verfahren unter anderem wegen der Verknüpfung von Windows mit dem *Internet Explorer*, wobei die konkurrierende Unternehmung *Netscape* eine wesentliche Rolle spielte. Auf die Berufung von Microsoft folgte 2001/2002 die außergerichtliche Einigung.

Auf europäischer Ebene wurde das Verfahren im Jahr 2000 aufgrund der Beschwerde des mit Microsoft konkurrierenden Unternehmens *Sun Microsystems* eröffnet. Da es für IT-Märkte keine sektorspezifischen Gesetze gibt, welche national, wie etwa in den Bereichen Telekommunikation und Energie, den Zugang zu wesentlichen Einrichtungen regeln, fällt der Zugriff auf die Schnittstellendaten im Fall Microsoft grundsätzlich unter Art. 82 EG. Der Markt für PC-Betriebssysteme beziehungsweise deren Schnittstelleninformationen bildet dabei den vorgelagerten, der für Server–Betriebssysteme den nachgelagerten Markt, dessen Wettbewerb es zu schützen gilt.[6] Microsoft verweigerte ab der Einführung der Windows-Version 2000 den Zugang zu den Schnittstelleninformationen mit der Begründung, sie seien als geistiges Eigentum dem Zugriff anderer ohne entsprechende Lizenzen entzogen.

Microsoft wurde von der Kommission 2004 zur Offenlegung der Schnittstellen verpflichtet.[7] Gegen diese Verfügung hatte das Unternehmen zunächst Nichtigkeitsklage erhoben. Am 17. 9. 2007 hat das EuG die Entscheidung der Kommission u. a. in Bezug auf die Offenlegungsverpflichtung bestätigt. Microsoft hat sich inzwischen dieser Entscheidung gebeugt.

3.2 Die marktbeherrschende Stellung von Microsoft

Gemäß Art. 82 Satz 1 EG ist die missbräuchliche Ausnutzung einer beherrschenden Stellung auf dem Gemeinsamen Markt oder auf einem wesentlichen Teil desselben durch ein oder mehrere Unternehmen verboten, sofern dieses Verhalten zu einer

6 Siehe Stopper (2005, S. 87 ff.); die Kommission stellt dabei auf Arbeitsgruppenserver ab, siehe Fichert und Sohns (2004, S. 910).

7 Entscheidung abrufbar unter http://europa.eu.int/comm/competition/antitrust/cases/decisions/37792/en.pdf [06. Feb. 2008]; Zusammenfassung in den Pressemitteilungen der EU IP/04/382.

Beeinträchtigung des zwischenstaatlichen Handels innerhalb der Gemeinschaft führen kann. Art. 82 knüpft an eine bereits bestehende Machtposition an und verbietet deren missbräuchliche Ausnutzung, u. U. auch durch die Übertragung der marktbeherrschenden Stellung auf einen nachgelagerten Markt (Mestmäcker und Schweitzer 2004, § 15, Rn. 31 ff.).

Die marktbeherrschende Stellung von Microsoft auf dem Markt für Betriebssysteme ergibt sich aus der Natur dieser Systeme selbst. Da das Betriebssystem, vereinfacht ausgedrückt, als Vermittlung zwischen Hardware und Anwendungsprogramm funktioniert, ist Microsoft als führender Hersteller von Betriebssystemen in der Lage, sowohl über die Möglichkeit der Unterstützung von Hardware-Komponenten durch das eigene System als auch über die Weitergabe von Schnittstelleninformationen an Anwendungssoftwarehersteller zu entscheiden (Ishii und Lutterbeck 2002, S. 133).

Die marktbeherrschende Stellung von Microsoft ergibt sich außerdem durch den so genannten Netzwerkeffekt (Stopper 2005, S. 96): Durch die Kompatibilität der verschiedenen Microsoft-Systeme untereinander wurden die Produkte des Unternehmens mit wachsender Nutzerzahl immer attraktiver und die fehlende Kompatibilität mit Konkurrenzprodukten bündelte das Interesse der Verbraucher zusätzlich auf die Produkte von Microsoft. Aufgrund der verbreiteten Anwendung von Microsoft–Betriebssystemen stieg gleichzeitig die Nachfrage nach Server-Betriebssystemen anderer Hersteller, die fehlerfrei und ohne großen Aufwand mit Windows-Programmen zu verwenden wären. Der indirekte Netzwerkeffekt bewirkte daher, dass wegen der höheren Nachfrage bezüglich komplementärer Produkte der Anreiz auf Wettbewerber erhöht wurde, solche Produkte innerhalb ihrer Systeme anzubieten (Gallego 2006, S. 23).

In Anlehnung an den Fall *Bronner* verortete die Kommission die beherrschende Stellung von Microsoft auf einen potenziellen oder zumindest hypothetischen Markt. Es ging in jenem Verfahren um ein Hauszustellungssystem für Zeitungen als wesentliche Einrichtung.[8] Die Kommission hatte die Beurteilung, ob es mangels der Austauschbarkeit mit anderen Arten des Zeitungsvertriebs einen gesonderten, relevanten Markt für Hauszustellungssysteme gäbe, den nationalen Gerichten überlassen und war für die Bewertung des missbräuchlichen Verhaltens von einem hypothetischen Markt ausgegangen.[9] Microsoft brachte in diesem Zusammenhang vor, die Schnittstelleninformationen seien als solche mangels Bedarfs gar nicht existent, sondern müssten erst erstellt werden (Thyri 2005, S. 393); mit Blick auf die tatsächlichen wirtschaftlichen Gegebenheiten wird man aber mit der Kommission davon ausgehen können, dass entsprechend der Nachfrage der Wettbewerber von Microsoft ein potenzieller Markt für Schnittstelleninformationen besteht (Bartosch 2005, S. 245 f.). Dieses Argument war auch im Fall *Bronner* für ausreichend erachtet worden: Der Hauszustellungsservice wurde zwar nicht als gesondertes Produkt angeboten, war

8 EuGH, Urteil vom 26.11.1998, Rs. C-7/97, Slg. 1998, I-7791.

9 Siehe Rn. 34 f. des Urteils.

aber aus Sicht der konkurrierenden Verleger für den Vertrieb unerlässlich, sodass eine tatsächliche Nachfrage bestand.[10]

3.3 Rechtfertigung der kartellrechtlichen Zwangslizenz – eine Bewertung der Microsoft-Entscheidung

Microsoft hat im Wesentlichen anderen Softwareunternehmen Schnittstelleninformationen vorenthalten, welche für den Datenaustausch zwischen dem Windows–Betriebssystem und alternativer Software erforderlich sind. Dieses Verhalten ist aus unternehmensstrategischer Sicht verständlich. Es geht auch konform mit dem Willen des Gesetzgebers betreffend den Grundsatz der Vertragsautonomie und auch den umfassenden Schutz immaterieller Rechtsgüter. Zudem fügt sich das beschriebene Marktverhalten von Microsoft in das Wettbewerbsverständnis ein, welches die freie Marktwirtschaft letztlich prägt. Dies gilt auch für die Entscheidung Microsofts, die Freigabe der Schnittstelleninformationen zu einem Zeitpunkt einzustellen, in dem das Unternehmen seine vorherrschende Marktposition gerade erreicht hatte (Körber 2004, S. 882). Microsoft berief sich in Bezug auf die Schnittstellen zusätzlich auf Patentschutz, Copyright und die Wahrung wertvoller Betriebsgeheimnisse. Letztere ließ das EuG als unilaterale, unternehmerische Entscheidungen bei der Beurteilung völlig unberücksichtigt. Aber auch den Patentschutz als solchen sah das Gericht nicht als entscheidungserheblich an, da es für die Rechtmäßigkeit einer Zwangslizenz allein auf den Marktmachtmissbrauch ankäme.[11] Andernfalls sei die Anwendung von Art. 82 EG auf die Ausübung von Immaterialgüterrechten praktisch immer ausgeschlossen.[12]

Wesentliche Einrichtung

In der Praxis betrifft der Missbrauch einer marktbeherrschenden Stellung häufig gerade den Wettbewerb auf einem abgeleiteten Markt, denn das jeweils beherrschende Unternehmen setzt in dieser Situation seine Macht genauso ein wie auf nur einem relevanten Markt (Emmerich 2006, § 10, Rn. 8 ff.). Der Missbrauch einer marktbeherrschenden Stellung kann nämlich in der Verweigerung des Zugangs zu einer wesentlichen Einrichtung, wie z. B. einem Hafen oder auch ganzen Netzstrukturen, liegen,[13] wenn der Inhaber über diese Infrastruktur den Zugang zu vor- oder nachgelagerten Märkten beherrscht. Es sind dies wesentliche Einrichtungen oder Infrastrukturen, ohne deren Nutzung die Wettbewerber nicht am Wettbewerb auf einem vor- oder nachgelagerten Markt teilnehmen können und die sich auch nicht mit zumutbaren Mitteln duplizieren lassen (Körber 2004, S. 885).

10 EuGH, Gewerblicher Rechtsschutz und Urheberrecht (GRUR) 2004, S. 524, 527.
11 Microsoft-Entscheidung EuG, Rn. 278 ff.
12 Microsoft-Entscheidung EuG, Rn. 690 ff.
13 Bezug zur Microsoft-Entscheidung EuG, siehe dazu Heinemann (2006, S. 705 ff.).

Das auch unter dem Begriff Essential-Facilities-Doktrin bekannte Fallbeispiel steht somit für eine Wettbewerbsbeschränkung nicht nur im Bereich der wesentlichen Einrichtung selbst, sondern auch auf den vor- und nachgelagerten Märkten (Lange 2006, Hübschle, Rn. 976). Die Doktrin stammt ursprünglich aus der US-amerikanischen Gerichtspraxis. Rechtsquelle ist Sec. 2 des *Sherman Act* aus dem Jahr 1890. Diese Vorschrift verbietet, im Gegensatz zu Art. 82 EG und § 19 IV GWB, bereits das Bestehen beziehungsweise den Versuch der Errichtung eines Monopols.[14] Die Doktrin, welche in der amerikanischen Rechtsprechung unterschiedlich angewendet wurde und wird, ist von der Europäischen Kommission in verschiedene Entschließungen übernommen worden.[15] Eine wesentliche Einrichtung ist dort definiert als „eine Einrichtung oder Infrastruktur, ohne deren Nutzung ein Wettbewerber seinen Kunden keine Dienste anbieten kann".[16]

Eine Zwangslizenz nach Art. 82 EG ist danach nur dann gerechtfertigt, wenn der Zugang zu der wesentlichen Einrichtung für eine Teilnahme der Wettbewerber am nachgelagerten Markt unentbehrlich ist. Ob der Zugang zu den Schnittstelleninformationen für eine Teilnahme am Wettbewerb unentbehrlich ist, wurde in der Entscheidung des EuG v. a. auf die von Microsoft erhobenen Einwände hin behandelt. Microsoft hatte vorgebracht, Interoperabilität sei in abgeschwächter Form auch ohne die Freigabe von Schnittstellen erreichbar und im Übrigen bestehe bereits ein funktionierender Markt für Betriebssysteme anderer Anbieter.[17] Was die tatsächliche Gestaltung des Marktes angeht, sah sich das EuG nur eingeschränkt prüfungsbefähigt.[18] Er stellte jedoch fest, dass wegen des Umstands einer bereits existierenden Interoperabilität, welche es Wettbewerbern nicht ermöglicht, sich auf dem Markt für Server-Betriebssysteme zu behaupten (wie von der Kommission ausführlich dargelegt), auf einen ineffektiven Wettbewerb auf diesem Markt geschlossen werden könne.[19] Vor dem Hintergrund der Bronner-Entscheidung könnte man nach diesen Ausführungen zu der Beurteilung kommen, es sei eine – wenn auch weniger günstige – Zugangsalternative gegeben und somit bereits das Vorliegen einer wesentlichen Einrichtung ausgeschlossen, zumal auch ein Markt für so genannte Middleware bereits existiert. Der Europäische Gerichtshof (EuGH) hatte in seiner Entscheidung zum Fall *Bronner* betont, dass die Duplizierbarkeit der Einrichtung, zu der Zugang begehrt werde, besonders sorgfältig zu prüfen sei. Unentbehrlich ist die Einrichtung für den Marktzugang nach dieser Entscheidung nur dann, wenn keine (auch keine weniger günstigen) Zugangsalternativen bestehen oder geschaffen werden können. Jedoch ergibt sich schon aus der Systematik des Netzwerkeffektes, dass eine auch nur geminderte Interoperabilität den Marktwert eines Produkts und damit die Marktmacht des

14 Siehe dazu etwa Immenga und Mestmäcker (2007, Möschel, Art. 82, Rn. 239) mit weiteren Nachweisen.
15 Bspw. Hafen von Rodby, ABl. 1994, Nr. L 55, S. 52.
16 Sea Containers/Stena Sealink, ABl. 1994, Nr. L15, S. 8.
17 Microsoft-Entscheidung EuG, Rn. 343 ff.
18 Microsoft-Entscheidung EuG, Rn. 379.
19 Microsoft-Entscheidung EuG, Rn. 229.

herstellenden Unternehmens herabsetzt, da sich die Bindung größerer Nutzermengen gerade aus der unproblematischen Verbindung verschiedener Systeme untereinander ergibt. Im Ergebnis hat das EuG sowohl die marktbeherrschende Stellung als auch die Unentbehrlichkeit des Zugangs zu den Schnittstellen jedenfalls bestätigt.[20]

Der Ausschluss anderer, konkurrierender Unternehmen von den streitrelevanten Schnittstellen ist für sich genommen noch kein missbräuchliches Verhalten, auch wenn auf diese Weise die Übertragung von Marktmacht auf den nachgelagerten Markt überhaupt erst möglich wird (Immenga und Mestmäcker 2007, Möschel, Art. 82, Rn. 246 ff., 252). Hier ist zu berücksichtigen, dass der Ausschluss von Wettbewerbern seine erhebliche Wirkung erst dadurch entfalten konnte, dass zuvor die Marktmacht durch die Attraktivität eines kompatiblen Systems mit (damals noch) freien Schnittstellen und dem Zusammenhang mit der steigenden Nutzerzahl überhaupt erst geschaffen wurde (Thyri 2005, S. 396) und nicht durch gezieltes – und eben missbräuchliches – Verhalten von Microsoft. Insbesondere darf nicht bereits die marktbeherrschende Stellung von Microsoft Objekt der Missbrauchskontrolle sein, da der Netzwerkeffekt, d. h. die Aufwertung eines Produktes durch die hohe Anzahl seiner Nutzer, gerade ein Charakteristikum von IT-Märkten ist (Stopper 2005, S. 97).

In den Fällen, in denen Urheberrechte die wesentliche Einrichtung bilden, muss die Feststellung eines Missbrauchs jedoch noch strengeren Anforderungen genügen. Denn gerade durch das Schutzrecht wird dem Inhaber die Möglichkeit gegeben, Investitionen vorzunehmen in dem Bewusstsein, dass er ausschließlich berechtigt ist, die Vorteile daraus zu ziehen (Heinemann 2006, S. 705, 710 f.). Die Verweigerung einer Lizenz stellt nach einigen Urteilen des EuGH nur dann einen Missbrauch i. S. v. Art. 82 EG dar, wenn der Inhalt des Schutzrechts eine wesentliche Einrichtung darstellt und die willkürliche Lizenzverweigerung die Entstehung eines neuen Produktes zu Lasten des Verbrauchers verhindert sowie den Wettbewerb auf einem nachgelagerten Markt ausschließt.[21]

Verhinderung eines neuen Produktes

Probleme bei der Beurteilung des Microsoft-Falles wirft das im Fall *Magill* aufgestellte Erfordernis des neuen Produktes auf.[22] In *Magill* hatte der EuGH in seinem Urteil aus dem Jahr 1995 drei Voraussetzungen festgelegt, bei deren Vorliegen eine Lizenzverweigerung ausnahmsweise als missbräuchlich bezeichnet werden kann: Diese müsse das Auftreten eines neuen Produktes verhindern und im Übrigen nicht gerechtfertigt, sondern vielmehr geeignet sein, den Wettbewerb auf einem abgeleiteten Markt zu verhindern.[23] Streitgegenstand in *Magill* waren Informationen von Fernsehanstalten

20 Microsoft-Entscheidung EuG, Rn. 388 ff., 422.

21 EuGH Slg. 2004, I-5069 IMS NDC.

22 Das EuG nimmt auf das Kriterium als „besonderen Umstand" in Rn. 332 der Entscheidung erstmals Bezug.

23 EuGH, 6.4.1995, C-241 und C-242/91, Slg. 1995, I-743

über deren Programme, welche benötigt wurden, um aktuelle Programmzeitschriften zusammenzustellen.

Die Konkurrenten von Microsoft streben gerade danach, Produkte auf den Markt zu bringen, welche derselben Verbrauchererwartung entsprechen wie auch das Windows-System; andernfalls würden sie sich kaum um Interoperabilität bemühen. Der freie Zugang zu Schnittstelleninformationen ermöglicht Wettbewerbern höchstens die Entwicklung besserer Versionen, nicht aber völlig neuer Systeme. Schutzzweck von Art. 82 kann danach nur der Erhalt des Wettbewerbs auf dem bereits bestehenden nachgelagerten Markt sein (Stopper 2005, S. 102).

Die Kommission ging ebenfalls davon aus, dass das Kriterium des neuen Produktes nicht zu eng auszulegen sei. Vielmehr sei es ausreichend, dass die wesentliche Einrichtung unverzichtbar für weitere Innovationen ist, welche aber nicht bereits zu klar identifizierbaren Produkten herangereift sein müssen.[24] Nachvollziehbar ist jedenfalls die Argumentation der Kommission dahingehend, dass die Wettbewerber Microsofts, denen zuvor die Schnittstelleninformationen zugänglich waren, nicht den Markteintritt, sondern vielmehr die Marktbehauptung anstrebten, weshalb das strikte Festhalten am Kriterium des neuen Produktes nicht angebracht sei (Höppner 2005, S. 460).

Das EuG hat in seiner Entscheidung zu Microsoft auf die Argumentation der Kommission Bezug genommen: Ausreichend sei es, dass das Produkt des Wettbewerbers grundlegende Elemente enthalte, welche aus der eigenen Anstrengung des Wettbewerbers resultierten. Das EuG betonte, dass sich dies schon zwangsläufig so ergebe, da die Wettbewerber im Wege der Offenlegung der Schnittstellen lediglich deren Beschreibung, nicht aber deren Anwendung eröffnet bekämen. Sie könnten daher nur ihre eigenen Produkte verbessern, nicht aber die von Microsoft kopieren.[25] Zu solchen Verbesserungen wiederum seien sie auch gezwungen, um sich im Wettbewerb von ihren Konkurrenten positiv abzusetzen.[26].

Was im Übrigen die unterschiedliche Auffassung von Interoperabilität der Kommission auf der einen und Microsoft auf der anderen Seite angeht, so hat nach der Einschätzung des EuG Microsoft zwar behauptet, aber nicht ausreichend dargelegt, dass aufgrund der Offenlegungsverpflichtung Microsoft-Produkte von Wettbewerbern praktisch „geklont" werden könnten. Das Gericht schloss sich der Kommission an, die betont hatte, Wettbewerber sollten lediglich in die Lage versetzt werden, Produkte mit eigenen Funktionen und insbesondere auch eigenem Quellcode herzustellen, welche die „Sprache" des Betriebssystems von Microsoft verstehen können.[27]

24 Entscheidung der Kommission, Rn. 693 ff.; siehe hierzu kritisch: Körber (2004, S. 889 f.).

25 Microsoft-Entscheidung EuG, Rn. 631 ff., 639 ff.

26 Microsoft-Entscheidung EuG, Rn. 658.

27 Microsoft-Entscheidung EuG, Rn. 141, zum Wortlaut: Microsoft sollte *specifications* zur Beschreibung von *protocols* herausgeben, durch die aber gerade nicht auf den Quellcode zugegriffen werden kann.

Schutzzweck: Das Verbraucherinteresse

Die Abwägung zwischen der Entscheidungsfreiheit des Inhabers geistigen Eigentums im Umgang mit seinem Recht und dem Schutz der Wettbewerbsfreiheit kann nur zugunsten des Letzteren ausfallen, wenn anderenfalls eine Marktentwicklung zum Nachteil der Verbraucher verhindert würde. Das ergibt sich aus dem Schutzzweck des Art. 82 EG selbst.[28] Dies wiederum wäre dann der Fall, wenn die Herstellung oder der Vertrieb von Produkten, welche die Verbraucher nachfragen, ausgeschlossen würden. Generalanwalt Tizzano ließ es bereits im Fall *IMS Health* für die Möglichkeit einer Zwangslizenz ausreichen, dass durch das (verhinderte) Produkt eines Wettbewerbers „besondere Bedürfnisse" der Verbraucher befriedigt würden, auch wenn sich dieses Produkt im Verhältnis zu denen des Marktführers als Konkurrenz und nicht als völlige Neuerung darstelle.[29] Im Verfahren gegen die *IMS Health GmbH* 2004 war es um ein System zur Darstellung von Marktberichten über den Absatz von Arzneimitteln gegangen, welches sich zu einem De-facto-Standard entwickelt hatte.[30] Diesen Erwägungen ist ohne Weiteres zu folgen, denn wenn das Interesse der Wettbewerber das des Urhebers nur dann überwiegen kann, wenn andernfalls die Marktentwicklung zum Nachteil der Verbraucher behindert würde, ist der Begriff des neuen Produktes letztlich nicht statisch, sondern anhand des konkreten Verbraucherinteresses zu bestimmen.[31] Das EuG hat im Fall Microsoft einen, wenn auch nur indirekten, nachteiligen Effekt für die Verbraucher darin gesehen, dass diese mangels ausreichender Interoperabilität praktisch auf die Microsoft-Betriebssysteme beschränkt und damit von der Auswahl anderer, möglicherweise bevorzugter Systeme ausgeschlossen sind.[32]

3.4 Schlussfolgerung und Weiterentwicklung

Zu der Frage, ob die Verhinderung eines neuen Produktes zur Annahme eines missbräuchlichen Verhaltens nicht nur erforderlich, sondern bereits alleine ausreichend ist, trifft die Entscheidung des EuG keine klare Aussage. In letzterem Fall bestünde für ein marktbeherrschendes Unternehmen die Möglichkeit, durch die Herstellung oder Ankündigung des nachgefragten Produktes nicht nur den Missbrauchsvorwurf zu entkräften, sondern seine beherrschende Stellung auf den nachgelagerten Markt sogar noch auszuweiten. Dies würde jedoch eine Verstärkung des Monopols statt der durch Art. 82 EG bezweckten Beschränkung bedeuten. Umgekehrt kann nicht ohne Weiteres davon ausgegangen werden, dass das marktbeherrschende Unternehmen darauf verzichten wird, das neue Produkt entweder selbst auf den Markt zu bringen oder dies durch Lizenzvergabe zu ermöglichen, um sein Immaterialgüterrecht wirtschaftlich zu

28 Siehe oben, Abschnitt 2.
29 IMS Health, Kommission, ABl. 2002 L 59/18, Schlussanträge Tz. 62.
30 EuGH, 29.4.2004, C-418/01, RIW 2004, S. 620 ff.
31 IMS Health, GRUR 2004, S. 527, Rn. 48.
32 Microsoft-Entscheidung EuG, Rn. 652, 664.

verwerten.[33] Nur dann würde aber die Entwicklung des neuen Produktes tatsächlich verhindert werden.

Das Verbot missbräuchlichen Verhaltens in Art. 82 EG will den Wettbewerb auf dem jeweils relevanten Markt vor Verfälschungen schützen. Wenn in einem unverfälschten Wettbewerb diejenigen Anbieter am besten positioniert sind, die sich der Verbrauchererwartung am weitesten angenähert haben, muss Schutzzweck von Art. 82 EG also die Verhinderung einer für den Verbraucher nachteiligen Marktentwicklung sein. Das Kriterium des neuen Produktes macht vor dem Hintergrund des Schutzzweckes von Art. 82 EG daher nur dann Sinn, wenn darauf aufbauend die Verhinderung eines Wettbewerbs um das Produkt als Voraussetzung für die Zwangslizenz gefordert wird.

4 Fazit

Als Fazit wird man zunächst festhalten können, dass zumindest dem scheidenden Chef von Microsoft das Urteil des EuG offenkundig keine schlaflosen Nächte bereitet. Bill Gates ließ in seiner Abschiedsrede auf der *Consumer Electronics Show* in Las Vegas verlautbaren, dass sich nach wie vor die Mehrheit der Verbraucher für Microsoft-Produkte entscheide, „obwohl es diese ganzen kostenlosen Programme gibt".[34] Microsoft zeigt zumindest nach außen hin Vertrauen in die breite technische Aufstellung des eigenen Unternehmens, trotz offenzulegender Schnittstellen und der weiter stattfindenden Fortentwicklung von Software im Open-Source-Bereich.[35]

Das Verfahren gegen Microsoft hat durch die Vorarbeit der Kommission und die Entscheidung des EuG zwar die Voraussetzungen für eine kartellrechtliche Zwangslizenz präzisiert, jedoch eng begrenzt auf den konkreten Streitgegenstand. Von allgemeingültigen Tatbestandsvoraussetzungen oder gar einer Faustregel für die Abgrenzung von Kartell- und Urheberrecht generell kann dagegen nicht die Rede sein. Die in dem Verfahren zugrunde gelegten Rechtfertigungsvoraussetzungen für die Zwangslizenz bilden jedoch eine geeignete Ebene für Schlussfolgerungen im Hinblick auf jene Abgrenzung.

Auf die Frage „Was hat es gebracht?" wird man daher antworten können, dass zukünftige Entscheidungen, die in ähnlich gelagerten Fällen auf die Argumentation im Fall Microsoft, v. a. betreffend des Kriteriums des neuen Produktes, Bezug nehmen, nicht nur unverfälschten Wettbewerb im Sinne des Kartellrechts, sondern durch die geförderte Produktselektion seitens der Verbraucher diesen zumindest eine indirekte Mitgestaltung moderner Software ermöglichen können.

33 So auch Mestmäcker und Schweitzer (2004, Ullrich/Heinemann, Immaterialgüterrecht (GRUR), Rn. 59).

34 Frankfurter Allgemeine Zeitung vom 9.1.2008, S. 9, 13.

35 Frankfurter Allgemeine Zeitung vom 9.1.2008, S. 13.

Literatur

Bartosch, A. (2005), 'Der Zugang zu einer wesentlichen Einrichtung – eine Zwischenbilanz nach dem Beschluss des EuG-Präsidenten vom 22.12.2004 in der Rechtssache Microsoft', *Recht der Internationalen Wirtschaft* **51**(4), S. 241–246.

Bunte, H.-J. und Langen, E. (2006), *Kommentar zum deutschen und europäischen Kartellrecht*, Bd. 2, 10. Aufl., Luchterhand, München.

Emmerich, V. (2006), *Kartellrecht*, 10. Aufl., C. H. Beck, München.

Fichert, F. und Sohns, A. (2004), 'Wettbewerbsschutz auf dem Markt für Server-Betriebssysteme', *Wirtschaft und Wettbewerb* **54**(9), S. 907–917.

Freyermuth, G. S. (2007), Offene Geheimnisse – Die Ausbildung der Open-Source-Praxis im 20. Jahrhundert, *in* B. Lutterbeck, M. Bärwolff und R. A. Gehring (Hrsg.), 'Open Source Jahrbuch 2007 – Zwischen freier Software und Gesellschaftsmodell', Lehmanns Media, Berlin, S. 17–57.

Gallego, C. B. (2006), 'Die Anwendung des kartellrechtlichen Missbrauchsverbots auf „unerlässliche" Immaterialgüterrechte im Lichte der IMS Health- und Standard-Spundfass-Urteile', *Gewerblicher Rechtsschutz und Urheberrecht - Internationaler Teil* **55**(1), S. 16–28.

Heinemann, A. (2006), 'Gefährdung von Rechten des geistigen Eigentums durch Kartellrecht? Der Fall „Microsoft" und die Rechtsprechung des EuGH', *Gewerblicher Rechtsschutz und Urheberrecht* **108**(9), S. 705–713.

Hirsch, G., Montag, F., Säcker und Jürgen, F. (2007), *Münchener Kommentar zum Europäischen und Deutschen Wettbewerbsrecht (Kartellrecht)*, Bd. 1, C. H. Beck, München.

Höppner, T. (2005), 'Missbräuchliche Verhinderung „neuer" Produkte durch Immaterialgüterrechte', *Gewerblicher Rechtsschutz und Urheberrecht - Internationaler Teil* **54**(6), S. 457–464.

Immenga, U. und Mestmäcker, E.-J. (2007), *Wettbewerbsrecht*, Bd. 1 EG, Teil 1 und 2, 4. Aufl., C. H. Beck, München.

Ishii, K. und Lutterbeck, B. (2002), Der Microsoft-Prozess, *in* A. Roesler und B. Stiegler (Hrsg.), 'Microsoft. Medien – Macht – Monopol.', Suhrkamp, München, S. 130–153. edition suhrkamp 2281.

Jaeger, T. und Metzger, A. (2006), *Open Source Software. Rechtliche Rahmenbedingungen der Freien Software.*, C. H. Beck, München.

Körber, T. (2004), 'Geistiges Eigentum, essential facilities und „Innovationsmissbrauch" – Überlegungen zum Microsoft-Fall im Lichte der EuGH-Entscheidung IMS Health GmbH', *Recht der Internationalen Wirtschaft* **50**(12), S. 881–891.

Lange, K. W. (2006), *Handbuch zum deutschen und europäischen Kartellrecht*, 2. Aufl., Recht und Wirtschaft, Frankfurt am Main.

Mestmäcker, E.-J. und Schweitzer, H. (2004), *Europäisches Wettbewerbsrecht*, 2. Aufl., C. H. Beck, München.

Schricker, G. (2006), *Urheberrecht*, 3. Aufl., C. H. Beck, München.

Stopper, M. (2005), 'Der Microsoft-Beschluss des EuG', *Zeitschrift für Wettbewerbsrecht* **3**(1), S. 87–110.

Thyri, P. (2005), 'Immaterialgüterrechte und Zugang zur wesentlichen Einrichtung – Der Fall Microsoft im Licht von IMS-Health', *Wirtschaft und Wettbewerb* **2005**(4), S. 388–399.

Kapitel 3

Von der digitalen Herausforderung zum sozialen Prozess

„Unsere Einstellung der Zukunft gegenüber muß sein: Wir sind jetzt verantwortlich für das, was in der Zukunft geschieht."

– *Sir Karl Raimund Popper*

Open Source trägt Verantwortung

RICHARD BRETZGER

(CC-Lizenz siehe Seite 281)

Treffen ökonomische Anstrengungen und soziale Bewegungen zusammen, kommt es oft zu unbeabsichtigten Nebeneffekten. Während ein Großteil der Open-Source–Bewegung von der Öffentlichkeit noch als stark von sozialen Beweggründen getrieben betrachtet wird, treten ökonomische Ansätze der Open-Source-Entwicklung vermehrt auch auf dem Markt auf und scheinen ein eigenes Regelwerk der Marktökonomie zu entwickeln. Genau an dieser Stelle treffen nun zwei verschiedene Sichtweisen aufeinander: die der Ökonomen und die der Soziologen. Das darf natürlich nicht zu der Interpretation verleiten, hier stießen konträre Positionen zusammen. Gleichwohl sind die Perspektiven unterschiedlich: Die Akteure bedienen sich jeweils eines eigenen Wissensvorrats und spezifischer Methoden bei der Betrachtung des gemeinsamen Gegenstands.

Open Source war jedoch bereits von Anfang an von beiden Seiten getrieben, ökonomische Interessen waren nicht ohne soziale Beweggründe durchzusetzen, soziale Prozesse konnten sich nur durch überzeugende Ergebnisse behaupten.

Bei dem Versuch, beide innerhalb dieses Komplexes wirkende Kräfte isoliert zu betrachten, stellt sich unwillkürlich die Frage nach den Handlungs- und Entscheidungsgrundlagen, die den jeweiligen Interessen zugrunde liegen. Wie kommt es z. B. dazu, dass sich ein Unternehmen wie *IBM*, das in der Computerwelt als Manifestation der Profitgier und Konservativität gilt, an einer von freiheitlicher Selbstbestimmung und Wissensfreiheit getriebenen Bewegung beteiligt (vgl. Perens 2007, S. 132)? Nach welchen Kriterien werden Entwickler in den Produktionsprozess von Software einbezogen? Der Organisationssoziologe Herbert Simon zeichnet in seinem Werk *homo rationalis* verschiedene Vorstellungen von Rationalität auf und zeigt die Begrenztheit der Rationalitätsgrundlagen der Entscheidungen von Individuen. Der Konsequenz dieser *bounded rationality* folgend, lässt sich also kein berechenbares Muster, lassen sich keine eindeutigen Regeln aufstellen, anhand derer die Folgen von Entwicklungen und Prozessen in der Open-Source-Landschaft abschätzbar wären.

Gleichermaßen sind jedoch die Wirkungen avancierter Daten- und Informationstechnologie unmittelbar in der Gesellschaft zu spüren (vgl. Rammert 2007, S. 37 ff.).

Die Entwickler und Nutzer dieser Technologie sind auch Träger des sozialen, kulturellen und ökonomischen Kapitals (im Sinne Bourdieus) und tragen somit Verantwortung für ihre Entscheidungen und deren Folgen. Verantwortung zu tragen bedeutet jedoch auch Wissen über den eigenen Kontext hinaus zu akkumulieren – diesen Beitrag leisten im diesjährigen *Open Source Jahrbuch* Autoren aus dem Gebiet der Soziologie mit einem breiteren Blick, der nicht zu so scharfen Implikationen neigt und heuristische Zugänge ermöglicht.

Den Wandel der *Free- und Open-Source-Software*-Gemeinschaften untersucht im Folgenden *Andrea Hemetsberger*, indem sie die Kulturen, die sich im sozialen Diskurs entwickelt haben, untersucht und Gegensätze innerhalb verschiedener Bewegungen als Motor der nachhaltigen Diskussion hin zur gesellschaftlichen Verantwortung herausarbeitet.

Mit den immateriellen Gütern in virtuellen Arbeitsumgebungen befasst sich *Udo Thiedeke* in seinem Beitrag zu den medialen Bedingungen der Knappheitskommunikation. Er verdeutlicht die Auswirkungen der veränderten medialen Situation auf soziale Bedingungen der Gesellschaft.

Wie werden Entwickler eingebunden und wer entscheidet in den Entwicklergemeinschaften? Mit dieser Frage beschäftigen sich *Daniel Tepe* und *Andreas Hepp*, indem sie die deterritoriale Vergemeinschaftung in den digitalen Produktionsgemeinschaften untersuchen.

Bislang wenig in den wissenschaftlichen Diskurs gestellt sind Prozesse des Lernens in Open-Source-Softwareprojekten. Dies möchte *Christoph Koenig* in seinem Artikel verändern, indem er drei Ebenen von Lernprozessen entwirft und bildungstheoretisch untersucht.

Literatur

Bourdieu, P. (1992), Ökonomisches Kapital - Kulturelles Kapital - Soziales Kapital [Erstausgabe 1983], *in* P. Bourdieu (Hrsg.), 'Die verborgenen Mechanismen der Macht', VSA, Hamburg.

Perens, B. (2007), Open Source – ein aufstrebendes ökonomisches Modell, *in* B. Lutterbeck, M. Bärwolff und R. A. Gehring (Hrsg.), 'Open Source Jahrbuch 2007 – Zwischen freier Software und Gesellschaftsmodell', Lehmanns Media, Berlin.

Rammert, W. (2007), Die technische Konstruktion als Teil der gesellschaftlichen Konstruktion der Wirklichkeit, *in* W. Rammert (Hrsg.), 'Technik - Handeln - Wissen. Zu einer pragmatistischen Technik- und Sozialtheorie', VS Verlag für Sozialwissenschaften, Wiesbaden.

Simon, H. (1993), *homo rationalis: die Vernunft im menschlichen Leben*, Campus, Frankfurt am Main.

Vom Revolutionär zum Unternehmer – Die F/OSS-Bewegung im Wandel

ANDREA HEMETSBERGER

Die Free- und Open-Source-Software-Gemeinschaften (F/OSS) sind vielfach als neue soziale Bewegung beschrieben worden. In diesem Beitrag wird der Frage nachgegangen, welche Kulturen sich im sozialen Diskurs entwickelt haben, die die ideologischen Fundierungen der Bewegung widerspiegeln und die Bewegung nachhaltig aufrechterhalten. Dabei tauchen große Widersprüchlichkeiten innerhalb der F/OSS-Bewegung auf; die einen sind ideologisch–revolutionär, die anderen sind einem romantischen, weltverbessernden Ideal zugetan. Eine Gruppe ist radikal pragmatisch und unternehmerisch, andere sind skeptisch und warnen vor den Mechanismen kapitalistischer Marktstrukturen. Es sind aber gerade diese Gegensätze, die eine nachhaltige Diskussion provozieren, die F/OSS-Gemeinschaft am Leben erhalten und damit eine andere Art des Wirtschaftens und Miteinanderarbeitens für die Zukunft sicherstellen.

Schlüsselwörter: Soziale Bewegung · sozialer Diskurs · Unternehmertum · Ideologie

1 Einleitung

Viel hat sich getan. Eine Revolution ist gelungen. Nicht nur Insider und Protagonisten der Free- und Open-Source-Software (F/OSS) und ihrer Programmierer haben enthusiastisch über den Wandel von einer unternehmensgesteuerten, proprietären Softwareentwicklung hin zu einem globalen, offenen und selbst bestimmten Umgang mit Software berichtet. Auch in der akademischen Literatur sind F/OSS-Gemeinschaften vielfach als neue soziale Bewegung beschrieben worden. Tatsächlich hat sich durch deren Engagement unser Verständnis darüber, wie mit intellektuellem Kapital umgegangen werden soll, wer an neuen Produkt- und Softwareentwicklungen beteiligt

und wem der Output gemeinsamer Bemühungen zugänglich sein soll, radikal verändert. Bereits sehr früh haben unternehmerische Köpfe wie Tim O'Reilly (*O'Reilly & Associates, Inc.*) oder Bob Young (*Red Hat*) die F/OSS-Bewegung in eine unternehmerische Richtung geführt und damit zu weltweiter Verbreitung beigetragen. Mit Hilfe der *Open-Source-Initiative* haben auch manche Unternehmen sehr rasch erkannt, dass eine Zusammenarbeit mit der F/OSS-Bewegung zukunftsweisend sein kann und sich zu Partnern der Bewegung gemacht. Diese Tatsache hat in der F/OSS–Gemeinschaft immer wieder Anlass zu teils heftigen Diskursen und Anschuldigungen gegeben. Vertreter der radikal freien Bewegung warfen gemäßigteren Gruppen vor, sie würden die Idee der freien Software an proprietäre und machtgierige Konzerne verraten. Befürworter der pragmatischen Seite wiederum wehren sich gegen diese Vorwürfe und kämpfen für eine weltweite Anerkennung und Durchsetzung ihrer Idee.

Diese Diskussionen sind auch der wissenschaftlichen Forschung über soziale Bewegungen nicht neu. Dennoch haben weder Beteiligte noch Forscher bisher eine Antwort darauf gefunden, wie die F/OSS-Revolution möglicherweise enden wird. Abgesehen von soziologischen Betrachtungen, die sich aus theoretischer Sicht die Frage stellen, ob das F/OSS-Netzwerk überhaupt als eine soziale Bewegung betrachtet werden kann (Zimmermann 2004) oder nach welchen grundlegenden ideologischen Hintergründen sich die Bewegung entwickle (Weber 2004), steht nun die Frage im Raum, wie sich das große, heterogene und sich teils widersprüchlich artikulierende Online-Netzwerk entwickeln wird. Wird die einstmals gefeierte freie Bewegung von den dominanten Machtstrukturen des Markts einvernommen werden? Werden F/OSS-Projekte in Unternehmen mit kapitalistischen Strukturen enden, die sich von *Microsoft* et al. nur durch ihre Labels und Logos unterscheiden? Oder gelingt es, die ursprünglichen Zielsetzungen der Bewegung in eine erneuerte Sichtweise eines moralischen Unternehmers zu kanalisieren? In diesem Artikel wird der Frage nachgegangen, wie die F/OSS-Bewegung ihre revolutionär-unternehmerische Identität selbst intern verhandelt.

2 Die F/OSS-Gemeinschaft als aktive soziale Online-Bewegung

Während viele soziale Bewegungen aus einer gesellschaftlichen *Zwangs*lage, Stigmatisierung oder Marginalisierung heraus erklärt werden (können), ist im Fall der F/OSS–Bewegung eine mindestens gleich stark emanzipierte, *befreite* Form der Bewegung erkennbar. Sozialer Protest äußert sich nicht ausschließlich in einem Bedürfnis, jene Strukturen zu zerstören, die die Freiheit des Einzelnen und der Gesellschaft einschränken, und alternative Strukturen zu propagieren. Die F/OSS-Bewegung hat ihre Macht zumindest teilweise dadurch erlangt, dass sie keine antikapitalistische oder marktfeindliche Bewegung darstellt, sondern vielmehr durch kollektives Handeln Möglichkeiten aufgezeigt und geschaffen hat, die radikal alternative Formen der Zusammenarbeit, Produktion und Kreation von wertvollen öffentlichen Gütern hervorbringen. Es geht

nicht darum, etablierte Formen des Markts zu zerstören, sondern eine Alternative zu verwirklichen, die den ideologischen, aber auch individuellen Motiven der F/OSS–Entwickler und -Unterstützer entspricht und die Marktmachtverhältnisse zu ihren Gunsten zurechtrückt. Dazu bedient sich die F/OSS-Bewegung einer mächtigen globalen Netzwerkstruktur – des Internets –, auf die sie nicht nur frei zugreifen kann, sondern die sie auch technisch mitgestaltet.

F/OSS-Gemeinschaften arbeiten allesamt fast ausschließlich über das Internet. Das Netz bietet sozialen Bewegungen dabei einen kollaborativen Raum, der noch weitgehend geschützt ist von den Sphären des Markts und andererseits einen enorm großen Pool an intellektuellem und sozialem Kapital bereitstellt. Diese Tatsache hat sich die F/OSS-Bewegung zunutze gemacht und sich eine der wertvollsten Ressourcen unserer Zeit gesichert – Wissen und Know-how. Selbst die Tatsache, dass dieses Wissen in unterschiedlichsten Formen frei zugänglich ist und von jedermann eingesehen werden kann, trägt der Tatsache nicht Rechnung, dass die der F/OSS-Bewegung eigene Kultur des *sharing* (Berners-Lee 2006) und des Enthusiasmus, der in die gemeinsamen Projekte gesteckt wird, erst die Basis für die fruchtbringende *Verwertung* der kollektiven Expertise darstellt. Was überdies dazu kommt ist, dass reine Information und kodierbares Wissen nur eine Seite der Medaille darstellen. Die Genialität von gemeinsamen Projekten steckt in der kollektiven Erarbeitung neuer Ideen und Lösungen durch Menschen, die ein gemeinsames Verständnis von der Sache entwickelt haben; und das kann nur sehr schwer nachgeahmt werden.

Die Mobilisierung dieser Ressourcen über das Internet ist grundsätzlich einfacher, vorausgesetzt, die ideologische Basis stimmt. Machtstrukturen und sozialer Status stellen sich im Internet anders dar und basieren auf individuellen Beiträgen, die ins Netz gestellt und damit geteilt werden. Was nicht im Netz sichtbar wird, ist nicht vorhanden; was im Netz nicht vorhanden ist, kann nicht Basis von Macht oder Status sein. Raymond (1999) drückt es in ähnlicher Weise aus, wenn er sagt, dass sich Macht im Internet nicht auf jene Dinge stützt, die anderen Menschen vorenthalten und künstlich verknappt werden, sondern auf das, was mit allen geteilt wird. Damit entsteht ein neuer *Reichtum* an Wissen und Produkten, die durch soziale Netzwerke erarbeitet werden und Menschen auf der ganzen Welt zugute kommen (Benkler 2006). Auf diese Weise werden alternative Marktstrukturen geschaffen, anstatt alte Strukturen zu zerstören. So sozialromantisch dieses neue alte Modell sozialer Produktion auch klingen mag, so stellt sich im Gegenzug sofort die Frage, wie die Bereitschaft der Menschen aufrechterhalten werden kann, einen Beitrag zur F/OSS-Bewegung zu leisten – wie die ideologische Basis geschaffen werden kann, die für die Nachhaltigkeit solcher Netzwerke sorgt.

3 Kulturelle Codes im sozialen Diskurs

Melucci (1996) behauptet, dass soziale Bewegungen ihre ideologische Fundierung durch die Etablierung kultureller Codes schaffen. Um der kapitalistischen Logik des Markts zu entfliehen und alternative Strukturen aufzubauen, werden Gegenideologien aufgebaut und im sozialen Diskurs im Zeitablauf für derart selbstverständlich genommen, dass sie irgendwann gleich plausibel erscheinen wie die Ursprungsideologien. Was in der physikalischen Welt als gegeben und dominant erscheint, kann der Einzelne im Netz aufbrechen, sich in gleichberechtigter Art und Weise an Austausch, Wertkreation und Produktion beteiligen und sich dadurch von der Passivität des Lebens als Arbeitnehmer und Konsument emanzipieren. Damit verliert der Markt im Internet teilweise seine ideologische Kraft und muss zumindest einer gleichberechtigten netzbasierten peer-to-peer-Struktur weichen. Der Aufstieg und die Entwicklung von *Napster* (Giesler 2006) legt Zeugnis über diese Entwicklung ab. Diese Entwicklung entfaltet aber erst ihre Kraft im sozialen Diskurs, im Entstehen einer gemeinsamen Kultur, Sprache und Andersartigkeit im Denken und Handeln.

Wenn Menschen ihre eigene Welt konstruieren, so verwandeln sie das Fremde in Vertrautes und legen sich eine verständliche Sprache und Praktiken zurecht, die ihrer Welt entsprechen. So hat die F/OSS-Bewegung zum Beispiel die für uns als selbstverständlich angenommenen Copyrights auf den Kopf gestellt und *Copyleft-Lizenzen* daraus gemacht. *Copylefted* bedeutet, dass die Software rechtlich davor geschützt ist, dass jemand, der die Software weiterentwickelt oder in seine Programme einbindet, ein Copyright auf das neue Programm erheben kann. Das führt dazu, dass Softwarehersteller, die *copylefted* Software in ihre Software eingebaut haben, dazu gezwungen sind, ihre neue Software auch unter der *Copyleft-Lizenz* herauszugeben. Ihre kreative Sprachwahl und der radikale Einsatz der *Copyleft-Lizenzen* haben ihre Sprache und die damit verbundenen Praktiken bald zu etwas ganz Normalem gemacht: Im Netz ist man einfach offen. Solche Sprachen und Praktiken einer sozialen Bewegung sind ganz zentral in ihrer Funktion als Grenzziehung zu dem, was bisher war, was die Mitglieder der Bewegung ablehnen, was sie nun anders denken und wie sie anders handeln wollen. Erst die Unterscheidung von dem, was anders ist, macht eine soziale Gruppe zu einer unterscheidbaren Gruppe und verleiht ihr Identität. Die Unterscheidungsmerkmale werden durch so genannte kulturelle Codes im sozialen Diskurs entwickelt und weitergetragen. Kulturelle Codes bündeln eine Vielzahl von Unterscheidungen (Giesen 1999). Sie spiegeln die ideologische Fundierung einer Bewegung wider und halten sie nachhaltig aufrecht. Dies schafft Identität und ein Gefühl der Gemeinsamkeit.

Nun möchte man glauben, dass die Kraft dieser kulturellen Codes durch Einheit geschaffen wird. Zahlreiche Autoren belegen allerdings, dass die F/OSS-Kultur durchsetzt ist von teils dialektischen und widersprüchlichen Diskursen und Denkhaltungen. Es stellt sich die Frage, ob und wie diese Widersprüchlichkeiten gelöst werden (müssen) oder ob sie sogar eine wichtige Rolle für die Nachhaltigkeit des F/OSS–

Systems spielen. Um darauf eine Antwort zu finden, wurde eine Diskursanalyse von Postings auf *Slashdot.org*[1] und Reaktionen von der erweiterten F/OSS-Gemeinschaft der Jahre 1998 bis 2006 durchgeführt und die zu Tage tretenden kulturellen Codes wurden beschrieben. *Slashdot.org* ist eine moderierte Internetplattform, die 1997 gegründet wurde und traditionell von Softwareentwicklern, Computer-, Technologie- und Internetinteressierten bevölkert wird. Auf ihr finden zahlreiche Diskussionen zu mittlerweile fast 170 Themengebieten statt. Es finden sich darunter technische, wirtschaftliche, internetspezifische, aber auch gesellschaftspolitische Themen bis hin zu Spielen und Science Fiction. Jene Postings auf *Slashdot.org*, die im Jahresdurchschnitt die meisten Reaktionen hervorgerufen haben (also die meisten *replies* bekommen haben) wurden ausgewählt und daraufhin untersucht, welche kulturellen Codes der jeweilige Diskurs transportiert.

Die Analyse bringt zwei grundlegende Kategorien kultureller Codes zu Tage, die in ihrer Natur dialektisch sind und nach Giesen (1999) als traditionelle Codes und universalistische Codes bezeichnet werden können. Traditionelle Codes sind darauf gerichtet, einen klaren ideologischen Kern zu bewahren und in die Zukunft hinein zu sichern und weisen permanent auf die historischen Wurzeln der Bewegung hin. Traditionelle Codes schließen daher auch aus: das, was nicht sein soll; das, was zur Veränderung ins Gegenteil führen könnte; das, wogegen man sich wendet; das, was den Kern der Bewegung gefährdet. Universalistische Codes hingegen schließen ein. Integration, Offenheit für anderes – auch das vormals Feindliche –, Pragmatismus und Fortschritt prägen diesen Diskurs. Aufgrund des dialektischen Charakters dieser beiden Diskurse bleiben, trotz nicht zu vermeidender Entartungen in Form von *flames*[2], sowohl der ideologische Kern der Bewegung als auch der Fortschritt und Blick in die Zukunft im Diskurs der F/OSS-Bewegung erhalten. Diese Spannung im Diskurs erzeugt produktive Energie und Nachhaltigkeit.

3.1 Traditionelle Codes

Traditionelle Codes konfrontieren und zeichnen ein eher dogmatisches Bild von der Welt. Durch traditionelle Codes drückt die F/OSS-Bewegung aus, dass Revolution notwendig ist, um etwas zu erreichen. Gleichzeitig werden Animositäten gegenüber monopolistischen, kapitalistischen Unternehmen verteidigt; man kämpfe ja den Kampf des Guten, des *David gegen den Goliath*. Sich gegen Goliaths zu wehren, wird im Diskurs als selbstverständliche und offensichtliche Aufgabe dargestellt. Die Historie habe gezeigt, dass es notwendig sei, sich zu wehren, um die Menschheit/Software zu befreien und die Gesellschaft/Software zu verbessern. Zu diesem Zweck wird symbolisch immer wieder der Gegner, das Böse, artikuliert und klar herausgestellt, wer auf welcher Seite steht. Eine weitere wichtige kulturelle Funktion traditioneller Codes

1 Siehe http://slashdot.org.

2 Unter *flame* wird allgemein ein ruppiger oder polemischer, in jedem Fall unsachlicher Kommentar verstanden.

in der F/OSS-Gemeinschaft besteht darin, jene Taten verbal zu sanktionieren, die von der revolutionären Zielsetzung der F/OSS-Bewegung abweichen. Ein berühmtes Beispiel dafür ist die Anschuldigung Richard Stallmans, dem wohl berühmtesten Vertreter der Ideologie der freien Software und Gründer der *Free Software Foundation*, gegenüber der *KDE*-Gemeinschaft, die zur Programmierung eine *library*[3] verwendet, welche anfangs nicht unter der *GNU General Public License* (*GPL*) stand. Diskurse, die traditionelle Codes transportieren, werden sehr evangelistisch und ambitioniert geführt. Traditioneller Diskurs ist lebhaft, provokativ und resultiert oft in *flames*. Das ist wichtig, denn traditioneller Diskurs hat die Aufgabe, den ideologischen Kern ständig neu aufzuladen und immer wieder neu zur Gemeinschaft stoßende User und Mitglieder zu sozialisieren.

3.2 Universalistische Codes

Die F/OSS-Bewegung hat auch universalistische Codes im sozialen Diskurs entwickelt, die die zentrale Funktion der Weiterentwicklung, des Fortschritts und der Integration haben. Universalistische Codes orientieren sich am Ziel des produktiven sozialen Wandels und arbeiten mit der Sprache der Vernunft. Ihr Hauptargument betont die freie Weitergabe von Code, Wissen und Ideen, ohne Altruismus zu predigen, sondern mit dem Versprechen der Befreiung und der Emanzipation durch *free sharing*. Indem universalistische Codes Integration, Aufklärung und Pluralismus predigen, bilden sie einen wichtigen Kontrapunkt zu den bewahrenden und revolutionären traditionellen Codes. Im Zeitablauf haben universalistische Codes stark an Bedeutung gewonnen und somit zum Wachstum der F/OSS-Bewegung beigetragen. Universalistische Codes bilden die Grundlage für eine nachhaltige Weiterentwicklung und ständige Erweiterung der Bewegung, die dazu geführt hat, dass die F/OSS–Bewegung heute auch eine wichtige wirtschaftliche Macht in der Softwareindustrie geworden ist. Durch ihren universalistischen Charakter tragen sie allerdings auch dazu bei, dass sich die Kultur der F/OSS-Bewegung stark in Richtung Mainstream bewegt. Durch die Offenheit gegenüber industriellen Partnern und auch durch eigene unternehmerische Tätigkeiten werden zunehmend Stimmen laut, die vor den Gefahren der leisen Einvernahme durch den kapitalistischen Markt warnen. Wird die ehemals revolutionäre Bewegung als friedlicher, profitorientierter *market player* enden?

4 Über den Wandel der F/OSS-Bewegung – der unternehmerische Revolutionär

Betrachtet man die Entwicklung der F/OSS-Bewegung von der nüchternen Seite und verfolgt man ihre Entwicklung vom Revolutionär zum (kompromittierbaren?) Unternehmer, so drängen sich weitere Fragen auf, die in Zukunft im sozialen Diskurs der

3 Gemeint sind Bibliotheken von Unterprogrammen zur Programmierung von Software.

Bewegung beantwortet werden müssen. Eine dieser zentralen Fragen lautet: Passen ihre ideologischen Grundfeste und ihr unternehmerisches Handeln in ein zukünftiges Vorstellungsbild der F/OSS-Bewegung? Oder wird die F/OSS-Gemeinschaft vielmehr im Lauf der Zeit vom Markt übernommen? Ein neuer Diskurs scheint sich zu manifestieren, der diese Fragestellungen seit geraumer Zeit mit einschließt. Utopisch–romantisch und libertär gefärbter Diskurs wird von pragmatischem und skeptischem Diskurs begleitet.

Utopisch-romantischer Diskurs war und ist immer noch geprägt von einem humanen Menschenbild, dem helfenden Mitglied der Gemeinschaft, das sein Wissen und seine Software schenkt, weil es selber so reich beschenkt wurde durch das Wissen und die Hilfe anderer. Diese stark ethische Komponente der F/OSS-Bewegung ist aber nicht nur durch leere Worthülsen proklamiert worden, sondern vor allem auch durch die radikal humanen Praktiken der Gemeinschaft. Die Forschung über soziale Bewegungen belegt, dass viele Gruppierungen daran scheitern, dass Worten keine Taten folgen (können). Nun fällt es aber einer Gemeinschaft wie der F/OSS-Gemeinschaft nicht schwer, herzugeben, wenn dadurch nichts zu verlieren ist. Was aber noch wichtiger scheint, ist die Tatsache, dass jene, die von den „Geschenken" der F/OSS–Bewegung profitieren, nicht in die Rolle derjenigen gedrängt werden, die nun in der Schuld der Gemeinschaft stehen – im Gegenteil. Je mehr sie sich beschenken lassen (F/OSS-Software downloaden und verwenden), umso mehr tragen sie zur Verbreitung der Ideen und Ideologien bei. Der Beschenkte leistet also sogar noch einen Beitrag zur Gemeinschaft. Dies unterscheidet Online-Gemeinschaften in radikaler Weise von anderen „Hilfsprojekten" und stärkt damit den aktiven Austausch zum Wohle aller.

Dies haben frühe Protagonisten immer schon kolportiert und sogar evangelistisch gepredigt. Es ist die Freiheit, die mit dieser Offenheit verbunden ist, das kapitalistische Denken umzudrehen und gerade durch scheinbar altruistische Handlungen umso mehr Erfolg und sogar Profit zu machen. Mehr noch: Es befreit von der Marktlogik, die uns alle zu entfremdeter Arbeit zwingt und fehlerhafte Software produziert – so die Freiheitsideologie der F/OSS-Bewegung. Die Funktion der Freiheitsideologie, die übrigens politisch alle Extreme mit einschließt, und die Funktion der Politik des offenen Zugangs zu Software, Wissen, neuen Freunden und Möglichkeiten zum Mitarbeiten liegen in der Hauptsache darin, Motivationen zur Mitarbeit anzusprechen. Dabei werden nicht nur altruistisch denkende Menschen mit eingeschlossen, sondern auch rein egozentrische Motivationen zugelassen, ohne das Gemeinsame zu verraten. So geschieht es auch, dass es F/OSS-Ersteinsteiger erstmal genial finden, dass sie alle Software gratis aus dem Netz laden können und damit sogar „herumspielen" dürfen. Erst später wird der Ehrgeiz stärker, etwas beitragen zu wollen, um etwas zurückzugeben und der Welt zu zeigen, was man alles kann. Eine größere Motivation, als seine Arbeit der ganzen Welt zeigen zu dürfen, gibt es übrigens kaum.

Aus dieser Tatsache hat sich ein zweiter utopisch-romantischer Diskurs entwickelt, begleitet von der Diskussion um die Möglichkeiten und Arbeitsbedingungen in der

heutigen spätkapitalistischen Zeit. Trotz enormer Entwicklungen in der heutigen Arbeitswelt, weg von einem tayloristischen Arbeitsprinzip hin zu mehr Unternehmertum und Selbstbestimmtheit in der Arbeit, wird von der F/OSS-Bewegung schamlos aufgedeckt, wie sehr selbst scheinbar arbeitnehmerfreundliche Maßnahmen letztendlich lediglich der Profitsteigerung seiner Erfinder dienen. Jobs in der Softwareindustrie werden von der Tätigkeit des Programmierens für die F/OSS-Gemeinschaft im Diskurs schärfstens unterschieden. Freies Programmieren ist endlich wieder Passion und (Handwerks-)Kunst, ein freiwilliges Eintauchen in Arbeit, die von Herzen kommt, Spaß macht, die man mit Freunden teilt und die die Welt verbessert. Dennoch flüchtet man sich nicht in eine ökonomische Utopie, sondern macht, was jeder gerne macht: Miteinander in passionierter Art und Weise zusammen etwas schaffen, das schöpferische Element spielerisch zum Vorschein bringen und stolz darauf sein. Zudem wird aus den Diskursen auch deutlich, dass die freie Weitergabe der Resultate von passionierter Arbeit eine viel intimere, persönliche Form der Weitergabe seiner Arbeit darstellt. Der Arbeitnehmer also, der zurückgefunden hat von entfremdeten Jobs hin zur bereichernden Tätigkeit?

Utopisch-romantischer Diskurs wird innerhalb der F/OSS-Gemeinschaft mit realistisch-pragmatischem Diskurs zurechtgerückt. Pragmatischer Diskurs macht darauf aufmerksam, dass es nicht nur um die Selbstverwirklichung des Einzelnen in der Arbeit geht, sondern darum, etwas zu tun, was gebraucht wird – und das in exzellenter Qualität. Realistisch-pragmatischer Diskurs steht in scharfem Gegensatz zu romantisierenden Ideen und schlägt einen anderen Weg vor, um die Welt und die Gesellschaft zu verbessern, nämlich Zusammenarbeit mit gutgesinnten Partnern aus der Industrie, Verbreitung der Ideen in den Köpfen von vormals kapitalistisch geprägten Unternehmern/Unternehmen und das Hervorbringen exzellent programmierter Software. Es geht um das Programmieren guter Software, nicht um das romantisierte Bild des Bastlers in der Stube. Die Freiheitsideologie wird im pragmatischen Diskurs zur *Freiheit zum Tun*, anstatt der *Freiheit von Zwängen*, passionierte Arbeit wird zur hohen Ingenieurskunst, Altruismus und Teilen wird zu Kooperation. Eine pragmatische Ethik hat keine scharfen Prinzipien, sondern fördert an die Situation angepasste Handlungen und Praktiken. Das eröffnet wiederum flexible Handlungsmöglichkeiten und entschärft Beziehungen mit Außenstehenden, was integrierend wirkt. Pragmatischer Diskurs lässt Kooperationen mit industriellen Partnern und unternehmerisches Denken zu.

Doch Partnerschaften mit der kapitalistischen Welt machen kompromittierbar. Dies ist zentraler Inhalt skeptischen Diskurses in der F/OSS-Bewegung. Scheinbarer weltweiter Erfolg und erfolgreiche Partnerschaften mit der Industrie könnten sich ins Gegenteil wandeln, die F/OSS-Bewegung könnte von den vorherrschenden Machtstrukturen des Markts vereinnahmt werden. Wie die Geschichte von Napster zeigt, ist die vorherrschende Marktstruktur sehr rasch imstande, ihre Interessen mit legistischen Systemen durchzusetzen. Was nicht sein darf, aber trotzdem vom Markt

gewünscht wird, wird von vorherrschenden Wirtschaftssystemen vereinnahmt und in verharmloster Form wieder in das kapitalistische Marktsystem eingebunden. Die F/OSS-Bewegung ist sich dessen immer mehr bewusst und setzt der pragmatischen Denke scharfe Skepsis entgegen. Ähnlich wie alle anderen sozialen Bewegungen hat auch die F/OSS-Gemeinschaft die Tendenz, einen Outsiderstatus aufrechterhalten zu wollen, um ihre Existenz als soziale Bewegung zu sichern und zu rechtfertigen (Potter 2004). Beinahe jede soziale Bewegung verteidigt diese elitäre Stellung, um glaubhaft eine soziale Bewegung mit Revolutionsanspruch zu bleiben. Im sozialen Diskurs zeigen sich diese Bemühungen durch das Aufdecken der subtilen Belohnungsmechanismen des Markts und des Geldes und ihren Zusammenhang mit Kompromisshandlungen der Gemeinschaft. Offensichtliche Übernahmen und Übernahmeversuche der Industrie, wie zum Beispiel *Novells* Kauf von *SUSE Linux*, lösen immer Diskurs und Gegenwehr aus. Skeptischer Diskurs geht aber weit darüber hinaus, versucht subtile Mechanismen aufzudecken und für eine nachhaltige Aufrechterhaltung des revolutionären Potenzials der F/OSS-Bewegung zu sorgen.

Die Vielfältigkeit des Diskurses innerhalb der F/OSS-Gemeinschaft zeigt uns, dass die Online-Welt nicht nur einen offenen Zugang zu Software und Wissen bedeutet, sondern auch die Ansammlung unterschiedlichster Meinungen und Ideologien, die Mitglieder der Gemeinschaft in Bezug auf die F/OSS-Bewegung haben können. Online-Bewegungen sind weit davon entfernt, einer Meinung zu sein oder zu werden. Was aber bedeutet das für die Nachhaltigkeit derartiger sozialer Bewegungen im Netz? Betrachtet man die Diskurse im Zeitablauf, so entdeckt man, dass durch entgegengesetzte Meinungen immer wieder Energie und Kraft im Diskurs entsteht. Es braucht scheinbar nicht nur einen externen Feind wie *Microsoft* oder neuerdings *Google*, sondern erst durch die interne Konfrontation schafft es die F/OSS-Gemeinschaft, eine ideologisch untersetzte, aber dennoch wirtschaftlich erfolgreiche Institution zu sein. Der Ausgang ist zwar ungewiss, doch es zeichnet sich ein kultureller Code ab, der die F/OSS-Bewegung oder das F/OSS-Unternehmen (?) im Diskurs als moralisch handelnden Unternehmer zu umschreiben versucht. Abbildung 1 fasst die Dimensionen dieser dialektischen Diskurse zusammen. Solange es diese Diskurse gibt, die immer wieder den ethisch-moralischen Kern der Bewegung energetisieren und gleichzeitig eine zukunftsorientierte, unternehmerisch-pragmatische Vorgehensweise proklamieren, wird sich die Gemeinschaft nachhaltig zu einem moralischen Unternehmertum bekennen und es auch praktizieren. Die Frage, ob man die F/OSS-Gemeinschaft noch als revolutionäre Bewegung einstufen kann oder nicht, wird damit nebensächlich.

5 Fazit

Die F/OSS-Gemeinschaft hat in den letzten Jahrzehnten viele wirtschafts- und gesellschaftspolitische Änderungen eingeleitet. Was die Bewegung so interessant macht, ist nicht so sehr ihr revolutionärer Charakter, sondern ihr produktiver und aktiver Zugang

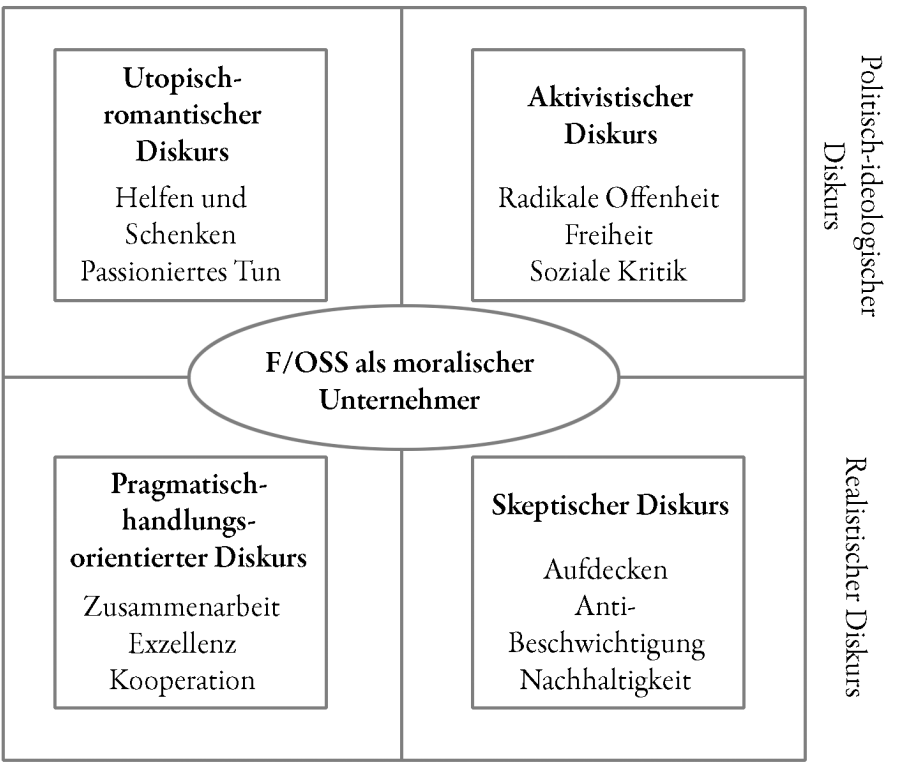

Abbildung 1: Dimensionen der dialektischen Diskurse

zu sozialer Veränderung. Mit dem weltweiten Zugang zum Internet wurde es erstmals möglich, soziale und intellektuelle Ressourcen im großen Stil zu mobilisieren. Durch die Einbindung einer globalen Entwicklergemeinschaft in eine bereichernde Form der selbst gesteuerten Arbeit und eine andere, unabhängige und zukunftsorientierte Form der kollektiven Produktion und Innovation hat die Gemeinschaft eine andere Art des Wirtschaftens und Miteinanderarbeitens aufgezeigt; eine, die noch dazu kreativer und schneller ist als herkömmliche Formen tayloristischer Arbeitsteilung in management-gesteuerten globalen Unternehmen. Basis dieses Erfolgs ist vielleicht, dass nicht die kapitalistische Form des Markts in Frage gestellt wurde, sondern deren fehlgeleitete Auswüchse. Damit hat sich die F/OSS-Gemeinschaft nie vom Markt abgekoppelt und ein Einzelschicksal gesucht, sondern das Funktionieren des Markts im Sinne al-

ler beteiligten Marktpartner ein Stück zurechtgerückt. Dass ihr dies bisher trotz der weltweit umspannenden und überaus heterogenen Gemeinschaft gelungen ist, ist bemerkenswert. Wesentlich dazu beigetragen hat eine Kultur des offenen Diskurses, der manches Mal schwer zu ertragen sein mag in seinen Windungen, Widersprüchen und Langatmigkeiten, aber oft auch sehr positiv und fruchtbringend geführt wird und dadurch auch immer wieder emotionalisiert und energetisiert. Fast jeder kann mitreden und seine Meinung kundtun. Viele tun es auch. In diesem Sinne wird auch sehr viel *social noise* erzeugt, also öffentlich sichtbare Kommunikation zu einem bestimmten Meinungsgegenstand, der die F/OSS-Bewegung lebendig erhält. Möge diese Übung auch in Zukunft gelingen.

Literatur

Benkler, Y. (2006), *The Wealth of Networks – How Social Production Transforms Markets and Freedom*, Yale University Press, New Haven.

Berners-Lee, T. (2006), *Weaving the Web – The Past, Present, and Future of the World Wide Web by its Inventor*, Texere, London.

Giesen, B. (1999), *Kollektive Identität – Die Intellektuellen und die Nation 2*, Suhrkamp, Frankfurt.

Giesler, M. (2006), 'Gift Systems', *Journal of Consumer Research* **33**(2), S. 283–290.

Melucci, A. (1996), *Challenging Codes – Collective Action in the Information Age*, Cambridge University Press, Cambridge.

Potter, J. H. A. (2004), *Nation of Rebels: Why Counterculture became Consumer Culture*, Harper Business, New York.

Raymond, E. S. (1999), *The Cathedral and the Bazaar. Musings on Linux and Open Source by an Accidental Revolutionary*, O'Reilly & Associates, Sebastopol, CA.

Weber, K. (2004), Philosophische Grundlagen und mögliche Entwicklungen der Open-Source- und Free-Software-Bewegung, *in* B. Lutterbeck, R. A. Gehring und M. Bärwolff (Hrsg.), 'Open Source Jahrbuch 2004 – Zwischen Softwareentwicklung und Gesellschaftsmodell', Lehmanns, Berlin, S. 369–383.

Zimmermann, T. (2004), Open Source und Freie Software – soziale Bewegung im virtuellen Raum?, *in* B. Lutterbeck, R. A. Gehring und M. Bärwolff (Hrsg.), 'Open Source Jahrbuch 2006 – Zwischen Softwareentwicklung und Gesellschaftsmodell', Lehmanns, Berlin, S. 353–368.

Digitale Produktionsgemeinschaften: Open-Source-Bewegung als deterritoriale Vergemeinschaftung

DANIEL TEPE UND ANDREAS HEPP

Der gegenwärtige soziokulturelle Wandel ist unter anderem durch eine fortschreitende „Mediatisierung" geprägt: Medienkommunikation, gerade auch die mittels digitaler Medien, prägt zunehmend verschiedenste Bereiche des Alltagslebens. Dabei werden digitale Medien insbesondere für die kommunikative Teilhabe an Kommunikationsnetzwerken wichtig und hierbei für die gesellschaftliche Verortung des Einzelnen bzw. die Partizipation an Wissensressourcen. Dies zeigt sich exemplarisch an der Open-Source-Bewegung. Das Ziel des Aufsatzes ist es, die Open-Source-Bewegung als eine sich auf Medienkommunikation gründende translokale, d. h. ortübergreifende Produktionsgemeinschaft zu beschreiben und folgende Fragen aus einem kommunikations- und medienwissenschaftlichen Fokus zu beantworten: Wie konstituieren sich die Open-Source-Netzwerke kommunikativ im Detail? Welche partizipativen Möglichkeiten entfalten sich für ein auch politisch zu verstehendes Engagement? Welche typischen Motive und Einstellungen teilen die Programmierer? Mit Blick auf die Strukturen der Open-Source-Bewegung versuchen wir abschließend, das Phänomen als deterritoriale Vergemeinschaftungsform in einen größeren Kontext einzuordnen und mit Fragen des soziokulturellen Wandels von Vergemeinschaftungen im Allgemeinen in Beziehung zu setzen.

Schlüsselwörter: Vergemeinschaftung · kommunikative Netzwerke · Motive · Einstellungen

1 Einleitung

Scheinbar über Nacht haben sich im Internet zahlreiche Formen computerbasierter Kooperationsnetzwerke gebildet, deren Mitglieder gemeinsam Inhalte erzeugen und

verbreiten. Die bis in die 1960er Jahre zurückreichende Open-Source-Bewegung kann als ein besonderes Beispiel für solche „special interest groups" (Beck 2005) verstanden werden, die weit über ihre Grenzen hinaus eine Wirkkraft entfaltet. So werden deren Softwareprodukte nicht nur (zunehmend) breit genutzt, auch findet sich das Open-Source-Grundprinzip, kollektiv erzeugtes Wissen in digitaler Form jedem frei zugänglich zu machen, in aktuellen partizipativen Projekten, wie z. B. *Wikipedia*, wieder. Dieser Hinweis macht bereits deutlich, in welchem Spannungsverhältnis die Open-Source-Bewegung als digitale Produktionsgemeinschaft zu verorten ist: Einerseits geht es um eine kooperative Softwareentwicklung innerhalb von sozialen Netzwerken. Andererseits versteht sich die Open-Source-Bewegung gleichzeitig als eine deterritoriale, politische Vergemeinschaftung mit weitergehenden Zielen. Durch dieses umrissene Spannungsverhältnis zwischen kooperativer Softwareherstellung und deterritorialer Vergemeinschaftung erscheint die Open-Source-Bewegung als ein exemplarisches Feld der kommunikations- und medienwissenschaftlichen beziehungsweise mediensoziologischen Auseinandersetzung mit digitalen Produktionsgemeinschaften in einer „postulierten Netzwerkgesellschaft" (Castells 2001; Hepp 2006*a*). An ihr werden sowohl die „Mediatisierung" (Krotz 2007) als auch „kulturelle Globalisierung" (Tomlinson 1999) wissensbasierter Produktionsformen greifbar, die sowohl auf Partizipation als auch politischen und sozialen Wandel zielen. Dies machen exemplarisch die unterschiedlichen Begriffe deutlich, die versuchen, die Open-Source-Bewegung zu fassen: Mal wird das Phänomen als „Szene heterogener Ingenieure" (Holtgrewe 2004), mal als „homogene Medialvergesellschaftung mit Gemeinschaftssemantiken" (Sebald 2005), mal als „soziale Bewegung im virtuellen Raum" (Zimmermann 2004) charakterisiert.

Ziel unseres Beitrags ist es, die Open-Source-Bewegung als eine sich auf Medienkommunikation gründende digitale Produktionsgemeinschaft zu beschreiben, die letztlich in einer deterritorialen Vergemeinschaftung mündet. Hierzu wollen wir in einem ersten Teil die Begriffe Open Source und freie Software überblicksartig umreißen. Der zweite Teil des Beitrags greift empirische Ergebnisse aus einer eigenen Fallstudie (Tepe und Hepp 2007) beziehungsweise aus anderweitigen Untersuchungen auf und beschreibt die Motive und Einstellungen einzelner Akteure sowie die strukturelle Zusammensetzung der gesamten Open-Source-Bewegung. Ausgehend hiervon wird in einem dritten Teil darauf eingegangen, inwieweit die Open-Source-Bewegung aus kommunikations- und medienwissenschaftlicher Sicht als eine deterritoriale, politisch orientierte Vergemeinschaftung verstanden werden kann.

Unsere Fallstudie stützt sich auf Auswertungen von 14 qualitativen Interviews, die mit internationalen Angehörigen der Open-Source-Bewegung geführt wurden. Erhoben wurde das Material zwischen 2002 und 2003 im Rahmen des Forschungsseminars „Medien und Globalisierung" von einer studentischen Arbeitsgruppe unter der Beteiligung von Bernhard Frey, Thomas Grund, Nin Jiang, Sabine Treinen und Luise

Zagst an der Universität Trier.[1] Ausgewertet wurde das Material mittels einer offenen Kodierung im Sinne der „Grounded Theory" (Glaser und Strauss 1998; Krotz 2005). Die Untersuchung versteht sich als eine explorative Studie zum Phänomen der Open-Source-Bewegung aus kommunikations- und medienwissenschaftlicher beziehungsweise mediensoziologischer Perspektive, deren Ergebnisse wir im Weiteren in Bezug auf andere Forschungen zur Thematik diskutieren.

2 Begriffsbestimmung Open Source und freie Software

Freie Computerprogramme auf der Basis von Open Source werden immer populärer. Die einst nur von und für Experten entwickelten Programme erweisen sich mehr und mehr als (mindestens) gleichwertige Alternativen zu kommerziellen Produkten etablierter Softwareanbieter. Der Grund für diesen Erfolg ist aber nicht allein darin zu sehen, dass die Computerprogramme kostenlos, d. h. ohne die Zahlung von Lizenzgebühren erhältlich sind. Ausschlaggebend ist vor allem, dass die Anwender von Open Source schon früh in den Entwicklungsprozess und die Produktionsgemeinschaft eingebunden werden. Damit vergrößert sich nicht nur der Personenkreis derer, die Software programmieren, sondern auch die Zahl der Anwender, die ihre Erfahrungen und Anforderungen direkt an die Entwickler weitergeben.

Blickt man zurück auf die Entstehungsgeschichte freier Software, stößt man unweigerlich auf den Namen Richard Stallman, der schon in den 1970er Jahren ein anerkannter Programmierer war und bis heute als einer der Gründungsväter der gegenwärtigen Freie-Software-Bewegung und des GNU-Projekts[2] gilt. Sein Ziel war es, mit der Entwicklung von GNU ein konkurrenzfähiges Betriebssystem auf Open-Source-Basis auf den Markt zu bringen, welches das damals weit verbreitete, aber kommerzielle UNIX-System ersetzen sollte. 1985 gründete Stallman die *Free Software Foundation*, „eine nicht-gewinnorientierte und (in einigen Ländern) als gemeinnützig anerkannte Nichtregierungsorganisation, die sich der freien (wie in Freiheit) Software widmet"[3]. Daneben entwickelte er die *GNU General Public License*, das bis dato weltweit meist verbreitete Lizenzmodell für Open-Source-Software (Koglin und Metzger 2004). Als weiterer Meilenstein in der Entwicklung der Open-Source-Community gilt die Etablierung des Open-Source-Begriffs mit der Gründung der *Open-Source-Initiative*[4] im Jahr 1998. Hintergrund war es, einen markttauglicheren Begriff im Gegensatz zu Free Software zu kreieren.

1 Das Forschungsseminar war ein im Rahmen der Aktivitäten der Forschungsgruppe Jugend- und Medienkultur von Andreas Hepp und Waldemar Vogelgesang an der Technischen Universität Ilmenau (Kommunikations- und Medienwissenschaft) beziehungsweise der Universität Trier (Soziologie) realisiertes Lehrprojekt, das sich mit dem Verhältnis des Wandels von Medienkommunikation und Prozessen der Globalisierung auseinandersetzte.

2 *GNU* steht für „GNU is Not UNIX".

3 Siehe http://www.germany.fsfeurope.org/ [20. Sep. 2007].

4 Siehe http://www.opensource.org/docs/definition.php [20. Sep. 2007].

Unterschiede zwischen den beiden Begriffen freie Software und Open Source lassen sich auf den ersten Blick nur schwer erkennen, denn beide greifen auf dasselbe Grundprinzip zurück, das in erster Linie auf der Offenheit beziehungsweise Verfügbarkeit des Quellcodes beruht. Während damit allerdings der Begriff Open Source in seiner Grundorientierung beschrieben ist, reicht dieser Verweis auf die freie Verfügbarkeit nicht aus, um freie Software als Teilaspekt der Open-Source-Bewegung zu bestimmen; hier liegt der Schwerpunkt eindeutig auf einer politischen und ethischen Ebene. Die Differenz zwischen dieser allgemeinen Bewegung und einer speziellen Teilbewegung liegt in der Akzentuierung: Während bei Open Source vor allem pragmatische Aspekte wie Nützlichkeit, Funktionalität, Zuverlässigkeit und Effizienz von Software betont werden, steht bei freier Software der urheberrechtliche Freiheitsgedanke im Vordergrund. Verkürzt lässt sich konstatieren, dass freie Software eine politische Philosophie verfolgt, Open Source eher als eine Entwicklungsmethodologie zu verstehen ist (Grassmuck 2002, S. 232). Oder um es mit Stallmans Worten zu formulieren:

> „Freie Software ist eine politische Aktion, die das Prinzip der Freiheit über alles andere stellt. Das ist der fundamentale Unterschied zu Open Source, das einen rein praktisch motivierten Weg darstellt, um Software zu schreiben. Open Source stellt nicht die Frage nach der Freiheit der Nutzer. Open Source hat keine Ideologie." (Richard Stallman, zitiert von King 1999)

3 Motive, Einstellungen und Strukturen der Open-Source-Bewegung

Diese definitorischen Feinheiten lassen bereits vermuten, dass die Meinungen und Einstellungen bezüglich der Begrifflichkeiten innerhalb der Open-Source-Bewegung alles andere als einheitlich sind. Und nach wie vor liefern diese Unterschiede immer wieder Anlass für Diskussionen unter den Vertretern der verschiedenen Lager, was letztlich auch auf unterschiedliche Motivlagen der Akteure des Open-Source-Netzwerks schließen lässt.[5] Vor diesem Hintergrund soll im folgenden Abschnitt genauer auf die Zusammensetzung der Bewegung eingegangen werden. Dabei stehen zwei Fragen im Vordergrund: Erstens, wer ist eigentlich die Community, die die Bewegung ausmacht? Und zweitens, welche Interessen und Motive haben die zum Teil hoch qualifizierten Programmierer, an Open-Source-Projekten teilzunehmen?

Aus – zumeist quantitativen – Studien liegen inzwischen zahlreiche Informationen über die sozialdemografische Zusammensetzung des Open-Source-Netzwerks vor. Die Ergebnisse zeigen, dass Programmierer durchschnittlich unter 30 Jahre alt und

5 Zur Vereinfachung wird im nachfolgenden Text nur noch von Open Source gesprochen und der Begriff freie Software nur an den Stellen verwendet, an denen die (wie auch immer gelagerte) politische Ebene explizit benannt werden soll.

männlich sind. Sie haben in der Regel einen Hochschulabschluss und arbeiten signifikant häufig im IT-Bereich (vgl. International Institute of Infonomics 2002 sowie Robles et al. 2001). Die Annahme, dass es sich bei Open-Source-Programmierern um jugendliche Freaks handelt, die sich in ihrer Freizeit gemeinsam gegen große Softwarekonzerne auflehnen, erweist sich auf der Basis des derzeitigen Kenntnisstands als Mythos. Dies ist insofern nicht überraschend, als inzwischen auch zahlreiche Softwarefirmen (wie z. B. *Red Hat* oder *Novell*) – die zum Teil aus Open-Source-Projekten entstanden sind oder ihre Firmenstrategie an Open-Source-Modelle angepasst haben – ihre Mitarbeiter für die Programmierung von Open-Source-Software bezahlen (Brand und Schmid 2004). Selbst Konzerne wie *Microsoft*, die proprietäre Software herstellen, haben die Vorteile der kollaborativen Entwicklung erkannt und versuchen, erfolgreiche Open-Source-Arbeitsmodelle in ihre Produktionsweisen zu integrieren (vgl. Hilf 2006).[6]

Hinsichtlich der Motivlagen der Mitglieder der Open-Source-Bewegung lässt sich auf der Basis der bestehenden Forschungsergebnisse konstatieren, dass zwar teilweise einige Verdienstmöglichkeiten – insbesondere für herausragende Programmierer – bestehen, die meisten Programmierer allerdings vor allem wegen Spaß am Programmieren selbst und Begeisterung für Technologie zu Open Source kommen.

Diesem Spaß am Programmieren liegt das Interesse zugrunde, sich gezielt mit komplexen Problemen zu beschäftigen und diese möglichst auch zu lösen. Für die Entwickler ist es – so einer unserer Interviewpartner – eine „persönliche Herausforderung" (Interview 8:7), auftretende Probleme in einem bestimmten Programm zu analysieren, das heißt die Fehler im Quellcode zu finden und dann die Codezeilen so umzuschreiben, dass das Programm letzten Endes funktioniert. Wenn man berücksichtigt, dass schon bei einem wenig umfangreichen Computerprogramm schnell einige hundert Zeilen Code zusammenkommen, wird die Fehlerbehebung schnell zu einer akribischen Suche beziehungsweise komplexen „Tüftelei" (Interview 8:8). Dabei eröffnen sich dem Programmierer aber auch Möglichkeiten, seine eigene Kreativität unter Beweis zu stellen, indem er versucht, die vorhandene Programmierstruktur zu optimieren, so dass das Programm am Ende noch besser funktioniert.

Wird das Interesse am Fehlersuchen und Verbessern von Quellcode von nahezu allen Programmieren geteilt – „we are all interested in the technical solutions to solve a problem" (Interview 6:1) – muss betont werden, dass die Herausforderung und

6 Ein Grund für das zunehmende Interesse von Anbietern kommerzieller Software an Open Source ist wohl auch mit dem gegenwärtigen Wegfall von Marktanteilen im Segment der Anwendersoftware verbunden. Dass Open-Source-Produkte nicht nur von Privatanwendern verstärkt genutzt werden, sondern gerade auch für den öffentlichen Sektor zunehmend interessant werden, wird an zahlreichen Fallbeispielen aus der Praxis aufgezeigt. Neben finanziellen Einsparungen aufgrund fehlender Lizenzgebühren sind es auch Gründe wie die langfristig zu erwartende Softwarevielfalt, die Plattform- und Herstellerunabhängigkeit und flexible Anpassungsmöglichkeiten an Firmenwünsche, die die Entscheidung zugunsten Open Source maßgeblich beeinträchtigen (Hoegner 2006; Stein und Zimmermann 2006).

Verantwortung mit der Entwicklung eines eigenen Programms bedeutend ansteigt. Die Veröffentlichung des eigenen Projekts bedeutet gleichzeitig, sich der Kritik anderer Programmierer aus der Community zu stellen. Dies geschieht in den meisten Fällen auf eine konstruktive Art und Weise, da lediglich die Qualität der Arbeit bewertet wird und nicht die Persönlichkeit des Programmierers. Somit machen für viele auch die Aussicht auf einen persönlichen Lernerfolg und die Möglichkeit, die eigenen Kompetenzen zu steigern, den Reiz am Open-Source-Programmieren aus.

Grundsätzlich ist es aber nicht ausgeschlossen, durch eine gute Arbeit Geld zu verdienen. So besteht für erfolgreiche Programmierer die Chance, durch ein Projekt von der Softwareindustrie entdeckt zu werden und einen Arbeitsplatz angeboten zu bekommen. Aber auch direkte Erlöse durch den Verkauf von Datenträgern mit Software und entsprechenden Zusatzleistungen sind möglich. Jedoch ist eine solche Wertschöpfung je nach Lizenzmodell an bestimmte Prämissen gebunden. So gestattet es die *General Public License (GPL)* inzwischen nicht mehr, dass allein die Software als solche verkauft wird. Allerdings können Personen für angebotene Service- und Supportleistungen im Rahmen der Software, wie z. B. für die Herstellung und den Verkauf von Handbüchern, Geld verlangen. Wichtig ist, dass die modifizierten Programmversionen stets weiterhin im Sinne der Lizenz allen zur freien Verfügung stehen.

Holtgrewe (2004, S. 347) spricht in diesem Zusammenhang von „interessanten Grauzonen zwischen Arbeit und Freizeit", da zahlreiche Entwickler angeben, nicht direkt für ihre Arbeit an einem Open-Source-Projekt bezahlt zu werden. Sie kommt zu dem Schluss, dass sich die Motive zwischen zwei Polen systematisieren lassen, die man wiederum zwei sozialen Typen zuordnen kann, nämlich dem des „Ingenieurs" und dem des „Aktivisten".[7]

Neben den schon genannten Motiven, wie der monetären Entlohnung, dem Bedarf, ein eigenes technisches Problem zu lösen, dem intrinsischen Vergnügen am Programmieren und dem Weiterentwickeln von persönlichen Kompetenzen, nennen Open-Source-Entwickler als Motiv immer wieder die in Aussicht gestellte Anerkennung von Gleichgesinnten (Hertel et al. 2003; Weber 2004, S. 135–149). Mit der Anerkennung geht in vielen Fällen gleichzeitig auch ein Reputationsgewinn innerhalb der Community einher und man steigt in der internen Hierarchie auf, womit wiederum bestimmte Privilegien verbunden sind. Die Anerkennung für eine herausragende Programmierarbeit stärkt darüber hinaus auch das persönliche Selbstwertgefühl, wie dieser Interviewpartner berichtet:

> „[. . .] knowing you fixed something up, knowing how to feature

7 Finck und Bleek (2006, S. 221 f.) kommen zu dem Ergebnis, dass Altruismus als Hauptmotiv in vielen Fällen überbetont wird, weisen aber gleichzeitig darauf hin, dass finanzielle Aspekte gerne unberücksichtigt bleiben und nach ihrer Einschätzung circa die Hälfte aller Entwickler für ihre Arbeit in irgendeiner Form entlohnt wird.

> that everyone wants it. . . that's pretty good. That makes you feel good. Because everyone benefits from you." (Interview 6:2)

Die erbrachten Leistungen werden über die Projektgrenzen hinaus deutlich innerhalb des Netzwerks kommuniziert. Dies geschieht zum einen über verschiedene Mailinglisten, in denen über die geleistete Arbeit berichtet wird, oder durch Auszeichnungen auf innerhalb der Community einschlägig bekannten Internetseiten (wie z. B. www.sourceforge.net). Preisverleihungen und Ehrungen finden darüber hinaus auch im Rahmen von Events statt, wie z. B. der *FOSDEM*[8], bei der zahlreiche prominente Akteure der Open-Source-Bewegung zusammentreffen. Die Wahrnehmung einer solchen Anerkennung ist durchaus ein weiterer Aspekt in den Motivlagen ihrer Beteiligten:

> „It's nice that other people appreciate and use your work. And it's very rewarding to see to be useful." (Interview 7:2)

Das sich durch solche geteilte Anerkennungen konstituierende Zugehörigkeitsgefühl, Teil einer Gemeinschaft zu sein und für diese einen zumeist unentgeltlichen Beitrag zu leisten, bewegt viele dazu, an Open-Source-Projekten teilzuhaben (vgl. Raymond 1997). Explizit macht dies folgender Interviewausschnitt:

> „I mean, you don't get any money for it of course but you develop friendships with other people and you get a sense of accomplishment and reward when your code is actually good, so that stimulates you to be more. . . I think that's about the main reason." (Interview 12:2)

Gleichwohl verlieren die meisten Entwickler trotz der persönlichen Profilierung die erbrachte technische Leistung nicht aus den Augen:

> „Certainly one gets a certain amount of ego-building from having ones peers admiring ones work. It's not about how clever I am as I get really good work. It's not that personal." (Interview 6:2)

Wurden bislang vor allem persönliche Motivationen angesprochen, muss dem hinzugefügt werden, dass es daneben kollektive politische Motive gibt, die vor allem innerhalb der Free-Software-Community als politischem Zweig der Open-Source–Bewegung zu erkennen sind. Diese sind gepaart mit einem diffusen, aus der „Hacker–Kultur" (Castells 2005, S. 52 f.) bekannten (konzern- und globalisierungskritischen) „politischen" Engagement (vgl. Eckert 1991).

Dreh- und Angelpunkt für das *politische Lager* ist die von Richard Stallman gegründete Free Software Foundation. Für die Mitglieder und Anhänger der Foundation gilt – durchaus im Sinne Max Webers wertrational – Freiheit als oberste Maxime und steht sozusagen über jeglichem Handeln. Ganz in diesem Sinne betont Stallman:

8 FOSDEM steht für „Free and open source software developers' european meeting".

> „‘Free software’ is a matter of liberty, not price. To understand the concept, you should think of ‘free’ as in ‘free speech’, not as in ‘free beer’.“ [9]

Das Grundprinzip der Redefreiheit aufgreifend fordern die Stallman-Anhänger, dass (auch) Informatiker das Recht haben sollten, sich frei von Hindernissen zu artikulieren und dazu auch bereits an anderen Stellen formulierte Ideen und Erkenntnisse aufgreifen dürfen sollten. Dieses „Grundrecht“ wird laut Stallman durch proprietäre Software verhindert, wodurch der freie Fluss von Informationen (aus kommerziellen Gründen) unterbrochen werde. Deshalb solle man sich für freie Software engagieren, um Alternativen zu den auf „unmoralischen“ Prinzipien beruhenden proprietären Programmen zu entwickeln. Denn je mehr freie Software auf dem Markt erhältlich ist und durch ihre Qualität überzeugt, desto weniger kommerzielle Produkte würden verkauft werden.

Obwohl die Vertreter von freier Software immer wieder betonen, dass es in erster Linie nicht um eine kostenlose Verbreitung von Programmen geht, kommt eben diesem Aspekt eine große Bedeutung für die postulierten politischen und ethischen Ziele zu. Denn aufgrund der fehlenden Lizenzgebühren eröffnen sich kostengünstige Alternativen für „ärmere Länder“ (oder auch staatliche Einrichtungen hierzulande). Darüber hinaus ermöglicht es die Offenheit der Quelltexte anderen Programmierern, Wissenschaftlern und sonstigen Benutzern, die Programme nach ihren Wünschen zu verändern und weiterzuentwickeln. Auf diese Weise könnten technologische Abhängigkeiten ärmerer Länder gegenüber reicheren verringert und ein reziproker Wissenstransfer zwischen verschiedenen Ländern ermöglicht werden.

Ein weiteres Leitmotiv – insbesondere für Anhänger der Freie-Software-Bewegung – stellt entsprechend die Möglichkeit dar, Alternativen zu proprietären Systemen zu entwickeln, damit einen gemeinnützigen Beitrag zu leisten und Chancen für Minderprivilegierte zu eröffnen:

> „It’s not just protest. It’s direct action. What we do, we don’t say down with proprietary software [. . .] but that’s not the main activity. The main activity is developing free software. [. . .] We’re making an alternative. . . .“(Richard Stallman, Interview 11:1)

Zusammenfassend können wir also festhalten, dass die Motive für eine Beteiligung an der Open-Source-Bewegung zwei Dimensionen haben. Dies ist erstens die *persönliche* Dimension. Hier geht es darum, sich ausprobieren zu können und Anerkennung für die eigene Arbeit zu erfahren. Solche Motive dominieren bei den „Ingenieuren“ der Open-Source-Bewegung. Daneben gibt es eine *subpolitische Dimension*, bei der es um die Realisierung einer Softwareproduktion jenseits einer proprietären Kontrolle durch Unternehmen geht, die zumindest prinzipiell jedem Zugang zur Softwarenutzung und

9 Siehe http://www.fsf.org/licensing/essays/free-sw.html [20. Sep. 2007].

Software(weiter)entwicklung eröffnet. Der Begriff des Subpolitischen hebt dabei in Anlehnung an Beck (1993, S. 156) darauf ab, dass an dieser Stelle Aktivitäten jenseits des formalen Systems von Staaten vorliegen, welche ebenfalls als politisch verstanden werden. Diese subpolitische Dimension dominiert bei den so genannten „Aktivisten". Diese beide Formen der Beteiligung bedürfen allerdings der technischen Möglichkeit einer ortsübergreifenden Partizipation.

Wie wir gesehen haben, steht der Ausdruck Open Source für die kooperative Herstellung und freie Verbreitung von Software-Produkten, an der weltweit vernetzte Programmierer teilhaben. Eric Raymond erklärt in seinem Aufsatz „Die Kathedrale und der Basar" den Erfolg von *Linux* und anderen Open-Source-Projekten in dieser bis dato einzigartigen Entwicklungsweise:

> „Die Linux-Gemeinde gleicht [. . .] einem großen plappernden Basar mit verschiedenen Tagesabläufen und Ansätzen (repräsentiert durch die Linux-Archive, in die jeder einbringen kann, was er will)."[10] (Raymond 1997)

Demnach kann an der Produktion prinzipiell jeder teilnehmen, der über ausreichende Programmierkenntnisse verfügt und bereit ist, Zeit und Arbeit in ein Projekt zu investieren. Auf der Basis freiwilliger Partizipation entstehen hunderte von Open-Source-Projekten, deren Beteiligte sich selten persönlich kennen und dennoch kooperativ zusammenarbeiten. Das Erstaunliche an diesem „Bazar-Prinzip" ist, dass es trotz fehlender formaler Richtlinien und der dezentralen Streuung von Kompetenzen in vielen Fällen erfolgreich ist. Als Begründung dafür wird Entwicklern an vielen Stellen eine hohe Selbstorganisationskompetenz attestiert, die Finck und Bleek (2006, S. 214) – in Anlehnung an Weber (2004, S. 132) – allerdings als „idealisierende" Unterstellung bewerten. Sie führen den Erfolg vieler Projekte stattdessen auf die informellen Planungsstrukturen und die geringe Kontrolle zurück – zwei für die Open-Source–Organisation charakteristische Merkmale. Die Autoren verdeutlichen weiter, dass es das typische Organisationsmodell bei der Open-Source-Entwicklung nicht gibt und auch die immer wieder betonte Offenheit und Gleichstellung der Mitglieder innerhalb eines Projekts so pauschal nicht zutreffe. Im Gegenteil weisen viele Projekte (vor allem größere) stark hierarchische Strukturen auf, in denen wenige Personen die wichtigen Entscheidungen treffen.

Großprojekte, wie z. B. die Programmierung eines Kernels oder Internetbrowsers, werden in der Regel in funktionale Einheiten untergliedert, die aufgrund ihres modularen Aufbaus ineinandergreifen. Jedes dieser Teilprojekte wird wiederum von einem

10 Im Gegensatz dazu erinnern „herkömmliche" Verfahren der Softwareentwicklung an „von einzelnen, erleuchteten Künstlern oder einer Handvoll auserwählten Baumeistern" geschaffene Produkte, die hinter gut verschlossenen Türen „Stein um Stein" zusammengebaut werden und nicht an die Öffentlichkeit kommen, bevor die Zeit nicht endgültig reif ist.

Kernteam geleitet und durch freiwillige Helfer unterstützt.[11] Zu ähnlichen Ergebnissen kommen auch Brand und Schmid (2004), die konstatieren, dass es einen kleinen Kern von Entwicklern gibt, denen aufgrund des größten Fachwissens Entscheidungs- und Delegationskompetenzen zugesprochen werden. Um diesen Kern herum versammeln sich zahlreiche weitere Personen, die mit kleineren Aufgaben betraut werden. Im äußeren Kreis stehen letztlich die Anwender, die die Produkte testen und Rückmeldungen bezüglich auftretender Fehler und individueller Anpassungswünsche geben. Die Möglichkeit, dass diese Nutzer selbst irgendwann in den Kreis der Programmierer aufgenommen werden, ist zwar grundsätzlich vorhanden, aber aus verschiedenen Gründen nicht die Regel.

Vor allem fehlt es diesen Nutzern an den notwendigen Programmierkenntnissen (Brand und Schmid 2004). Denn trotz der Offenheit des Quellcodes erfordert es ein hohes Maß an Kompetenz, diesen auch zu verändern beziehungsweise eigene Funktionen in bestehende Programme zu implementieren. Priddat und Kabalak sprechen deswegen von einem „Open-Source-Klub", der in doppelter Hinsicht elitäre Strukturen aufweist:

> „Man grenzt sich gegenüber den Inkompetenten 'draußen' ab und innerhalb kann man sich gegenüber den Anonymen abgrenzen, indem man Leistungen zeigt, die andere akzeptieren. Der erste Statusgewinn ist die Anerkennung als Experte im Open-Source-Netzwerk. Damit wird die Mitgliedschaft ausgerufen. Aber erst, wenn man als produktiver Autor im Open-Source-Projekt hervortritt und andere das als ein besonderes Ereignis bestätigen, beginnt die Statushierarchie zu arbeiten: die Hochwertung gegenüber Nur-Mitgliedern." (Priddat und Kabalak 2006, S. 114)

Hinsichtlich des strukturellen Aufbaus der Open-Source-Bewegung lässt sich also festhalten, dass es sich bei Open-Source-Projekten, die ihre Arbeit in hierarchischen Systemen koordinieren und sich zum größten Teil selbst organisieren, um ein Zusammenfließen zahlreicher individueller Kompetenzen und Teilleistungen handelt. Mitentscheidend für das Funktionieren ist es, dass jeder Einzelne (zumindest in der Selbstwahrnehmung) selbst entscheidet, was er tun will und was nicht. Hier spiegelt sich auch das schon mehrfach erwähnte Freiheitsideal wider. An dieser Stelle werden Fragen der Struktur auf einer grundsätzlicheren Ebene relevant, nämlich Fragen nach der Struktur von Prozessen der Kommunikation im kollaborativen Arbeitsprozess, der die Zusammenarbeit in der Open-Source-Bewegung kennzeichnet (ausführlich in Tepe und Hepp 2007).

11 Es sei an dieser Stelle angemerkt, dass das Alter eines Programmierers kein Kriterium für seine Stellung innerhalb eines Teams ausmacht. Was zählt, sind lediglich Programmierkenntnisse, Engagement sowie (Selbst-)Organisations- und Kommunikationskompetenzen.

4 Die Open-Source-Bewegung als deterritoriale Vergemeinschaftung

Vor dem Hintergrund unserer Darlegungen aus einer kommunikations- und medienwissenschaftlichen Perspektive schlagen wir vor, die Open-Source-Bewegung insgesamt als eine deterritoriale (politische) Vergemeinschaftung der Softwareentwicklung zu beschreiben.

Der Begriff der deterritorialen Vergemeinschaftung hebt darauf ab, dass mit der Globalisierung von Medienkommunikation die Relevanz von translokalen Vergemeinschaftungen zugenommen hat, also solcher durch Prozesse der Medienkommunikation vermittelter Vergemeinschaftungsnetzwerke, die gerade nicht – wie die Gemeinschaft der Nation – territorial bezogen sind. Mit Vergemeinschaftung bezeichnen wir – in Anlehnung an klassische Überlegungen von Weber (1972, S. 21) – diejenigen sozialen Beziehungen, die auf subjektiv gefühlter Zusammengehörigkeit der Beteiligten beruhen. Entsprechend sind unter deterritorialen Vergemeinschaftungen solche Vergemeinschaftungen zu verstehen, die sich als Netzwerk subjektiv gefühlter Zusammengehörigkeit translokal über verschiedene Territorien hinweg erstrecken (Hepp 2006*b*, S. 282; Hepp in Druck).

Und exakt dies trifft für die Open-Source-Bewegung zu: Gerade auch durch die Nutzung von digitalen Kommunikationsmedien hat sich hier nicht nur ein kooperativer Arbeitszusammenhang konstituiert, wie man ihn beispielsweise für global agierende Konzerne konstatieren kann, sondern darüber hinaus ein Gemeinschaftsnetzwerk mit einer geteilten (sub)politischen Identität. Mit anderen deterritorialen Vergemeinschaftungen[12] teilt die Open-Source-Bewegung folgende drei Aspekte:

Netzwerke lokaler Gruppen Deterritoriale Vergemeinschaftungen artikulieren sich zuerst in lokalen Gruppen, die durch eine entsprechende Face-to-Face-Kommunikation gekennzeichnet sind. Diese verschiedenen lokalen Gruppen fügen sich zu einem übergreifenden translokalen Netzwerk. Diesen Aspekt finden wir in der Open-Source-Bewegung in Ortsgruppen, die sich in verschiedenen Städten konstituiert haben und in denen sich die unterschiedlichen Entwickler (und auch User) treffen.

Translokaler Sinnhorizont Innerhalb von Netzwerken deterritorialer Vergemeinschaftungen besteht ein translokaler Sinnhorizont, d. h. eine gemeinsame Sinnorientierung, die diese Vergemeinschaftungen als solche begründet. Der translokale Sinnhorizont wird durch Prozesse medienvermittelter Kommunikation aufrechterhalten, seien dies Medien der personalen Kommunikation (beispielsweise

12 Deterritoriale Vergemeinschaftungen finden sich allerdings nicht nur im Bereich der Softwareentwicklung, sondern sind ein generelles Phänomen in Zeiten fortschreitender Globalisierung der Medienkommunikation. Weitere Beispiele wären Jugend-, Freizeit- und Populärkulturen, ethnische Vergemeinschaftungen der Diaspora, politische Vergemeinschaftungen sozialer Bewegungen oder religiöse Vergemeinschaftungen. Siehe dazu im Detail Hepp (2006*b*, S. 280–196).

Chats innerhalb des Netzwerks) oder der Massenkommunikation (z. B. Fanzines der deterritorialen Vergemeinschaftung). Bei der Open-Source-Bewegung wird insbesondere über entsprechende Websites beziehungsweise Social-Software-Angebote ein orientierender (sub)politischer Sinnhorizont kommuniziert.

Deterritoriale Erstreckung Wie der Name deterritoriale Vergemeinschaftung schon sagt, erstreckt sich deren translokales Netzwerk nicht einfach in einem spezifischen Territorium. Dies heißt nicht, dass es innerhalb von deterritorialen Gemeinschaften keine Nationalisierungen gäbe, im Gegenteil: Es lassen sich in deren Netzwerken durchaus nationale und regionale Verdichtungen ausmachen. Jedoch gehen deterritoriale Vergemeinschaftungen nicht in solchen territorialen Verdichtungen auf, wie auch ihr Sinnhorizont deterritorial besteht. Exakt dies wird anhand der Open-Source-Bewegung greifbar.

Diese weiter kontextualisierte Einordnung verdeutlicht, dass die Open-Source–Bewegung kein singuläres Phänomen ist, sondern sie sich in dem Moment, in dem man sie als *mehr* denn als eine reine Entwickler-Community begreift, in eine breitere Tendenz der Herausbildung von medienvermittelten deterritorialen Vergemeinschaftungen einfügt. Ihre Besonderheit ist allerdings sicherlich darin zu sehen, dass sie wertrationale (sub)politische Aspekte deterritorialer Vergemeinschaftung mit dem zweckrationalen Aspekt der kooperativen und *freien* Softwareentwicklung verbindet.

Eine solche weiter kontextualisierende Einordnung erscheint einmal mehr sinnvoll, wenn man sie in Beziehung setzt zu weitergehenden Fragen des soziokulturellen Wandels. So hat Castells (2001) darauf hingewiesen, dass wir die Entwicklung des Internets nicht einfach als eine technologische Innovation begreifen können, die sich durchgesetzt habe. Vielmehr sollte diese als Ausdruck eines generellen und gerade erst beginnenden Wandels hin zu einer „Netzwerkgesellschaft" verstanden werden, also hin zu solchen Formen von Gesellschaft, in denen das Netzwerk das Prinzip der Organisation sozialer Beziehungen wird und andere Prinzipien – wie beispielsweise das geschlossener Organisationen – an den Rand drängt.

Greift man die Überlegungen Castells auf, so erscheint die Open-Source-Bewegung als gegenwärtige Manifestation der ursprünglichen „Kultur des Internets" (Castells 2005, S. 47). Diese ist für Castells durch eine spezifische Form der Wert- und Glaubensorientierungen gekennzeichnet, in deren Zentrum eine „Ideologie der Freiheit" (ebd.) steht. Stimuliert wurde dieser Freiheitsgedanke in erheblichem Maße durch die techno-meritokratische Kultur der frühen „Techno-Eliten" (Castells 2005, S. 49) des Internets. Diese steht in der „Gelehrten-Tradition des gemeinsamen wissenschaftlichen Unternehmens, der Reputation wissenschaftlicher Exzellenz, des peer review und des freien Zugangs zu allen Forschungsergebnissen bei gleichzeitiger Anerkennung der Urheber einer jeden Entdeckung" (Castells 2005, S. 51). Es wird deutlich, in welchem Maße die Open-Source-Bewegung als Fortsetzung dieser ursprünglichen Wertorientierung erscheint, die insbesondere die Entwicklung des Internets geprägt

hat. In diesem Sinne hat die Open-Source-Bewegung bis heute eine Dimension des „Elitären".

Literatur

Beck, K. (2005), *Computervermittelte Kommunikation im Internet*, Oldenbourg, München.

Beck, U. (1993), *Die Erfindung des Politischen. Zu einer Theorie reflexiver Modernisierung*, Suhrkamp, Frankfurt a. M.

Brand, A. und Schmid, A. (2004), 'Fallstudie eines Open Source-Projekts. Arbeitspapier aus dem DFG-Projekt Struktur und Funktionsweise elektronischer Arbeitsmärkte'. http://www.soz.uni-frankfurt.de/arbeitslehre/pelm/docs/FALLSTUDIE%20Open%20Source%20V1.pdf.

Castells, M. (2001), *Der Aufstieg der Netzwerkgesellschaft. Teil 1 der Trilogie Das Informationszeitalter.*, Opladen.

Castells, M. (2005), *Die Internet-Galaxie*, VS, Wiesbaden.

Eckert, R. (1991), *Auf digitalen Pfaden. Die Kulturen von Hackern, Programmierern, Crackern und Spielern.*, Westdeutscher Verlag, Opladen.

Finck, M. und Bleek, W.-G. (2006), Mythen, Märchen, Missverständnisse, *in* B. Lutterbeck, R. A. Gehring und M. Bärwolff (Hrsg.), 'Open Source Jahrbuch 2006 – Zwischen Softwareentwicklung und Gesellschaftsmodell', Lehmanns Media, Berlin, S. 207–218.

Glaser, B. G. und Strauss, A. L. (1998), *Grounded Theory. Strategien qualitativer Forschung*, Huber, Bern.

Grassmuck, V. (2002), *Freie Software. Zwischen Privat- und Gemeineigentum*, Bundeszentrale für politische Bildung, Bonn.

Hepp, A. (2006*a*), *Netzwerk, Konnektivität und Fluss. Konzepte gegenwärtiger Medien-, Kommunikations- und Kulturtheorie*, VS, Wiesbaden.

Hepp, A. (2006*b*), *Transkulturelle Kommunikation*, UVK (UTB), Konstanz.

Hepp, A. (in Druck), Medienkommunikation und deterritoriale Vergemeinschaftung: Medienwandel und die Posttraditionalisierung von translokalen Vergemeinschaftungen, *in* R. Hitzler, A. Honer und M. Pfadenhauer (Hrsg.), 'Posttraditionale Gemeinschaften. Theoretische Bestimmungen und ethnographische Deutungen', VS, Wiesbaden.

Hertel, G., Niedner, S. und Herrmann, S. (2003), 'Motivation of software developers in open source projects: An internet-based survey of contributors to the Linux kernel', *Research Policy* **32**, S. 1159–1177.

Hilf, B. (2006), Einblicke in das Microsoft-Linux-/Open-Source-Software-Lab, *in* B. Lutterbeck, R. A. Gehring und M. Bärwolff (Hrsg.), 'Open Source Jahrbuch 2006 – Zwischen Softwareentwicklung und Gesellschaftsmodell', Lehmanns Media, Berlin, S. 31–44.

Hoegner, W. (2006), Das Projekt LiMux - Freie Software für die Münchener Verwaltungsclients, *in* B. Lutterbeck, R. A. Gehring und M. Bärwolff (Hrsg.), 'Open Source Jahrbuch 2006 – Zwischen Softwareentwicklung und Gesellschaftsmodell', Lehmanns Media, Berlin, S. 59–72.

Holtgrewe, U. (2004), Heterogene Ingenieure – Open Source und Freie Software zwischen technischer und sozialer Innovation, *in* B. Lutterbeck, R. A. Gehring und M. Bärwolff (Hrsg.), 'Open Source Jahrbuch 2004 – Zwischen Softwareentwicklung und Gesellschaftsmodell', Lehmanns Media, Berlin, S. 339–351.

International Institute of Infonomics (2002), 'Free/Libre and Open Source Software: Survey and Study. FLOSS.'. http://www.infonomics.nl/FLOSS/report [6. Feb. 2008].

King, J. (1999), 'Freie Software ist eine politische Aktion - Interview mit Richard Stallman'. http://www.heise.de/tp/deutsch/special/wos/6475/1.html [6. Feb. 2008].

Koglin, O. und Metzger, A. (2004), Urheber- und Lizenzrecht im Bereich von Open-Source-Software, *in* B. Lutterbeck, R. A. Gehring und M. Bärwolff (Hrsg.), 'Open Source Jahrbuch 2004 – Zwischen Softwareentwicklung und Gesellschaftsmodell', Lehmanns Media, Berlin, S. 293–304.

Krotz, F. (2005), *Neue Theorien entwickeln. Eine Einführung in die Grounded Theory, die Heuristische Sozialforschung und die Ethnographie anhand von Beispielen aus der Kommunikationsforschung*, Halem, Köln.

Krotz, F. (2007), *Mediatisierung: Fallstudien zum Wandel von Kommunikation*, VS, Wiesbaden.

Priddat, B. P. und Kabalak, A. (2006), Open Source als Produktion von Transformationsgütern, *in* B. Lutterbeck, R. A. Gehring und M. Bärwolff (Hrsg.), 'Open Source Jahrbuch 2006 – Zwischen Softwareentwicklung und Gesellschaftsmodell', Lehmanns Media, Berlin, S. 109–122.

Raymond, E. S. (1997), 'The Cathedral and the Bazaar'. http://www.catb.org/%7Eesr/writings/cathedral-bazaar/cathedral-bazaar/.

Robles, G., Scheider, H. und et al., I. T. (2001), 'Who Is Doing It? A research on Libre Software devel-opers'. http://widi.berlios.de [20. Jan. 2006].

Sebald, G. (2005), Vergesellschaftungsprozesse in der „Free/Open Source-Softwareentwicklung", *in* M. Jäckel und M. Mai (Hrsg.), 'Online-Vergesellschaftung? Mediensoziologische Perspektiven auf neue Kommunikationstechnologien', VS, Wiesbaden, S. 91–103.

Stein, S. und Zimmermann, B. (2006), Migration – Vom Wunsch zur Wirklichkeit. Einleitung, *in* B. Lutterbeck, R. A. Gehring und M. Bärwolff (Hrsg.), 'Open Source Jahrbuch 2006 - Zwischen Softwareentwicklung und Gesellschaftsmodell', Lehmanns Media, Berlin, S. 45–47.

Tepe, D. und Hepp, A. (2007), Digitale Produktionsgemeinschaften: Die Open-Source-Bewegung zwischen kooperativer Softwareherstellung und deterritorialer politischer Vergemeinschaftung, *in* C. Stegbauer und M. Jäckel (Hrsg.), 'Formen der

Kooperation in computerbasierten Netzwerken: Beispiele aus dem Bereich „Social Software“, VS, Wiesbaden.

Tomlinson, J. (1999), *Globalization and Culture*, University Of Chicago Press, Oxford.

Weber, M. (1972), *Wirtschaft und Gesellschaft. Grundriss der verstehenden Soziologie*, Mohr Verlag, Tübingen.

Weber, S. (2004), *The success of open source*, B&T, Oxford.

Zimmermann, T. (2004), Source und Freie Software – soziale Bewegung im virtuellen Raum?, *in* B. Lutterbeck, R. A. Gehring und M. Bärwolff (Hrsg.), ‘Open Source Jahrbuch 2004 – Zwischen Softwareentwicklung und Gesellschaftsmodell’, Lehmanns Media, Berlin, S. 353–368.

Kapitel 4

Von der Entscheidung zum Einsatz

„Zu mancher richtigen Entscheidung kam es nur, weil der Weg zur falschen gerade nicht frei war."

– *Hans Krailsheimer*

Auf der Suche nach Mitteln, die den Zweck heiligen

MARC SCHACHTEL UND MALTE SCHMIDT-TYCHSEN

(CC-Lizenz siehe Seite 281)

Seit der Veröffentlichung der *GNU General Public License (GPL)* 1989[1] hat sich rund um das Prinzip Open Source und freier Software einiges getan. Während Unternehmen noch vor nicht allzu langer Zeit vor der Frage standen, entweder Entwickler einzustellen oder eine Lösung einzukaufen, kann heute ein ganz anderer Weg gewählt werden. Open-Source-Software (OSS) hat ihren Platz auch in strategischen Entscheidungen gefunden, wobei das Sparen von Lizenzgebühren nicht unbedingt die Hauptrolle spielt.

In diesem Kapitel sollen genau die Entscheidungskriterien, die über Lizenzkosten hinausgehen, eine gewichtete Rolle spielen. Es bietet Unterstützung bei der Findung und Umsetzung der Entscheidung für ein Projekt.

Ein wichtiges Argument für die Migration auf OSS – gerade für öffentliche Verwaltungen – zeigt Jochen Günther, neben anderen, in seinem Artikel auf: *Open Source schafft Arbeitsplätze*. Günther liefert statistische Entscheidungsgrundlagen, die eine Entscheidung für OSS bekräftigen. Die Grundlage und vor allem die Motivation für den Einsatz von Open Source sind also gegeben. Günther zeigt auf, dass man mit dieser Entscheidung keinesfalls alleine ist. Nicht nur im öffentlichen Sektor ist der Trend zum Einsatz von OSS ungebrochen.

Die nächste Frage, die sich stellt, ist: „Was sind meine Alternativen? Welches Projekt lohnt sich?". Die Antwort auf diese Frage muss sicherlich individuell gefunden werden, eine Unterstützung bietet allerdings Jan Suhr mit seinem Artikel *Messung von Offenheit an IT-Artefakten – Der Information Technology Openness Benchmark*. Denn ist ein potenzielles Projekt gefunden und man hat sich implizit entschlossen, OSS zu verwenden, kann eine Aussage über die „openness" eines Projekts ein Projektkriterium sein.

Die verwendete Lizenz ist ein weiteres wichtiges Thema, welches immer mit freier Software verflochten ist. Die am häufigsten publizierte und auch genutzte Lizenz ist die GPL mit ihren Derivaten. Henriette Picot klärt in ihrem Artikel *Die deutsche Rechtssprechung zur GNU General Public License (GPL)* rechtliche Hintergründe und

1 Vgl. http://www.free-soft.org/gpl_history/ [06. Feb. 2008].

Fragen zur GPL mit praktischer Relevanz. Bei der Entscheidung, welches Projekt genutzt oder weiterentwickelt wird, sollte auch die Lizenz, unter der die Quelltexte bzw. andere Quellen stehen, eine Rolle spielen. Der Artikel konzentriert sich auf die Version 2 der GPL, geht allerdings auch auf die jüngst veröffentlichte 3. Version und die *Lesser GNU General Public License (LGPL)* ein.

Ganz konkret beleuchten Tobias Hauser und Andi Pitsch die Einsatzmöglichkeiten der Alternativen aus dem Open-Source-Lager, wenn es um ein Content-Management–System (CMS) geht. Open-Source-Content-Management-Systeme sind inzwischen zu ernsthaften Konkurrenten ihrer Kollegen aus dem proprietären Lager herangewachsen. Praxisnah werden in *Open Source Content Management Eine kritische Betrachtung*, unter Berücksichtigung aller relevanten Aspekte, die Vor- und Nachteile der Alternativen aus dem Open-Source-Lager untersucht. Auch hier spielt die Einsparung durch den Wegfall der Lizenzkosten eher eine nachrangige Rolle.

Der Mensch ist ein Gewohnheitstier. Ist eine Umgebung erstmal angenommen, fällt es mitunter schwer, sich davon zu lösen. Das bezieht sich auch und gerade auf die Pixel, die wir tagtäglich auf unseren Bildschirmen betrachten. Mit der Entscheidung für OSS ist es also noch längst nicht getan. Jede Neuerung hat ihre Gegner. Für den Fall einer Desktop-Migration, d. h. weg von Windows, hin zu Linux auf den Arbeitsplatzrechnern, ist immer mit Widerstand der Menschen zu rechnen, die diese Umstellung verkraften müssen. Beate Groschupf und Natascha Zorn analysieren nicht nur die Hintergründe dieser Widerstände: *Change Management: Linux-Desktop-Migration mit Erfolg* zeigt Mittel und Wege auf, sie zu überwinden und die Entscheidung erfolgreich umzusetzen. Sie stellen dabei die Migration im Auswärtigen Amt als Praxisbeispiel vor.

Neben dem Auswärtigen Amt haben auch eine Reihe von Stadtverwaltungen bereits entschieden, ihre Arbeitsplätze nach OSS zu migrieren. Dafür gibt es gute Gründe. Welche bei den Entscheidern in diesen vier Städten konkret eine Rolle spielten und wie sie gewichtet wurden, präsentiert Mark Cassell in seinem Beitrag *Umstieg auf Open–Source-Lösungen in der Stadtverwaltung: Ein Vergleich der Städte Treuchtlingen, München, Wien und Schwäbisch Hall*. OSS schafft mehr Unabhängigkeit, welche in den vorgestellten Fällen für die Entscheidung zu ihrem Einsatz das wichtigste Argument ist.

Die Frage, welches Produkt, ob nun Open Source oder nicht, genutzt werden soll, hängt von vielen Kriterien ab. Die enthaltenen Artikel beleuchten diese Kriterien und sind als Ganzes eine Hilfestellung sowohl bei der Entscheidungsfindung als auch der Durchsetzung dieser. Schließlich sollte am Ende der Entscheidung auch wirklich der Einsatz stehen.

Die Bedeutung von Open Source in der öffentlichen Verwaltung und der IT-Branche

JOCHEN GÜNTHER

Der Einsatz von Open-Source-Software (OSS) im öffentlichen Sektor gewinnt immer mehr an Bedeutung. Trotz der großen Anzahl von Studien rund um das Thema Open Source wurde die Wechselwirkung zwischen IT–Anbietern und öffentlicher Verwaltung bisher jedoch kaum untersucht. Daher wurden vom *Fraunhofer IAO* einerseits öffentliche Einrichtungen als Anwendergruppe untersucht, in denen OSS immer mehr an Verbreitung gewinnt. Andererseits werden die Auswirkungen des immer stärker werdenden Einsatzes von OSS auf die Wertschöpfung in IT-Unternehmen analysiert. Die Studie soll darüber hinaus verdeutlichen, welchen Effekt der zunehmende Einsatz von OSS auf den „IT-Standort Deutschland" hat.

Schlüsselwörter: IT-Unternehmen · Studie · Wirtschaftliche Auswirkungen · Verwaltung

1 Einleitung

Im Jahr 2005 waren in der deutschen IT- und Kommunikationsbranche 774 400 Personen beschäftigt und erzielten einen Jahresumsatz von rund 134,1 Milliarden Euro.[1] Zum Vergleich: Die deutschen Unternehmen im Maschinen- und Anlagenbau erzielten im Jahr 2005 einen Umsatz von 151 Milliarden Euro.[2] Die IT- und Kommunikationsbranche ist damit einer der bedeutendsten Wirtschaftsbereiche Deutschlands. Einer der Wachstumsbereiche – interpretiert man die anhaltend positiven Marktsignale – war

1 Siehe http://www.bitkom.org/files/documents/ITK-Marktzahlen_Herbst_Kurzfassung_2007.pdf [13. Sep. 2007].

2 Siehe http://www.bmwi.de/BMWi/Navigation/Wirtschaft/Industrie/gesamtwirtschaftliche-bedeutung.html [13. Sep. 2007].

und ist der softwarebezogene Dienstleistungsbereich rund um Open-Source-Software (OSS).

Open Source gewinnt in der öffentlichen Verwaltung immer mehr an Verbreitung wie z. B. bei der spektakulären Entscheidung der Stadtverwaltung München bei der Migration von rund 13 000 Desktop-Rechnern und Anwendersoftware sowie den dazugehörigen Servern.[3] Dies zeigten auch viele weitere Projekte, wie die Migration der niedersächsischen Steuerverwaltung auf Linux[4] oder der Einsatz von *Thunderbird* und *Mozilla* bei der französischen Gendarmerie – um nur einige der prominentesten und aktuellsten Beispiele zu nennen.[5]

Die Vorteile, die aus dem Einsatz von OSS entstehen, werden von den Anwendern hauptsächlich in Kosten-, Sicherheits- und Stabilitätsvorteilen gesehen, aber auch in der Innovationskraft, gegeben durch die international verteilte Entwicklergemeinschaft, und in der freien Verfügbarkeit des in OSS enthaltenen Wissens.

Rund um den Themenbereich Open Source in der öffentlichen Verwaltung gibt es bereits einige Studien beziehungsweise Teilergebnisse von Studien, die vor allem den Verbreitungsgrad und Einsatzzweck von Open Source aufgezeigt haben. Die Wechselwirkung zwischen IT-Anbieterunternehmen und öffentlicher Verwaltung wurde in bisherigen Studien jedoch nicht untersucht.

Das *Fraunhofer-Institut für Arbeitswirtschaft und Organisation* (*IAO*) hat daher eine Trendstudie mit dem Titel „Open Source Software: Strukturwandel oder Strohfeuer" durchgeführt, die im Besonderen öffentliche Einrichtungen und IT-Unternehmen untersucht. Ziel der Studie ist es einerseits, öffentliche Einrichtungen als exemplarische Anwendergruppe, in denen OSS immer mehr an Bedeutung gewinnt (Gosh et al. 2002, S. 10), zu untersuchen. Andererseits sollen die Auswirkungen auf die Wertschöpfung von IT-Anbietern analysiert werden, die durch die Nachfrage der öffentlichen Einrichtungen hervorgerufen wird. Darüber hinaus soll die Studie verdeutlichen, welchen Effekt der zunehmende Einsatz von OSS auf den „IT-Standort Deutschland" hat und damit verbunden auch, welche Beschäftigungswirkung der Einsatz von OSS entfaltet.

Im Detail wurden im Rahmen der Studie aktuelle Daten in Bezug auf die tatsächliche Verbreitung, die Einsatzbereiche und Erfahrungen mit OSS bei IT-Anbietern und der öffentlichen Verwaltung erhoben. Ein bedeutender Aspekt der Studie ist dabei die Untersuchung der Auswirkungen des Einsatzes von OSS auf die Gestaltung und Effizienz interner Abläufe in öffentlichen Einrichtungen. Damit soll aufgezeigt werden, welche enormen Chancen der Einsatz von OSS bietet und welchen hohen Stellenwert OSS bereits jetzt schon bei IT-Verantwortlichen genießt.

Die Befragung, die Grundlage für die Studie „Open Source Software – Strukturwandel oder Strohfeuer?" war, erfolgte über das Internet mit Hilfe des *Fraunhofer Online*

3 Siehe http://www.heise.de/newsticker/meldung/48313 [13. Sep. 2007].
4 Siehe http://www.bundestux.de/themen/inl/145995.html [13. Sep. 2007].
5 Siehe http://www.bundestux.de/themen/inl/106315.html [13. Sep. 2007].

Assessment Tool. Sie wurde im Zeitraum vom 12. 06. 2006 bis zum 07. 07 .2006 durchgeführt. Potenzielle Interessenten und Teilnehmer an der Befragung wurden gezielt über Anschreiben auf die Befragung aufmerksam gemacht. Hierbei wurden für den Bereich der öffentlichen Verwaltung 2000 Einrichtungen der Größenklassen 1 bis 6 und Zugehörigkeit zu verschiedensten Trägerebenen in Deutschland angeschrieben. Bei den IT-Unternehmen dagegen wurden 5000 Unternehmen der unterschiedlichsten Größenordnung angeschrieben. In beiden Fällen wurden nur Entscheidungsträger wie Geschäftsführer oder IT-Verantwortliche zur Teilnahme aufgefordert.

Aufgrund der gewählten Datengewinnungsform über eine Online-Befragung kann nicht von einer klassischen Repräsentativität der im Folgenden gewonnenen Ergebnisse ausgegangen werden, jedoch gewährleistet die Zusammensetzung der Antworten eine gute Übertragbarkeit der Ergebnisse. Sowohl für die öffentlichen Einrichtungen wie auch für die IT-Unternehmen liegt die erzielte Rücklaufquote im üblichen Bereich.

2 Einsatz von Open Source in der öffentlichen Verwaltung

Im Bereich der öffentlichen Einrichtungen konnten insgesamt 115 gültige Antworten erzielt werden. Die Aufforderung zur Teilnahme erging bundesweit schriftlich an 2000 öffentliche Einrichtungen sowie Städte und Gemeinden der Größenklassen 1 bis 6.

Öffentliche Verwaltungen nach dem Verständnis dieser Studie sind Organisationen, die mit einem durch die Gesetzgebung bestimmten Auftrag betraut sind. Hierbei sind öffentlich-rechtliche Handlungsformen – über die verschiedenen Trägerebenen der öffentlichen Verwaltung wie Bund, Bundesländer und Kommunen – sowie privatrechtliche Handlungsformen einbezogen.

Bedingt durch einen zunehmenden Standortwettbewerb in einer globalisierten Welt, aber nicht zuletzt auch durch steigende Anforderungen von Seiten der Bürger an eine moderne Verwaltung, steht der ganze öffentliche Sektor unter einem enormen Veränderungsdruck. Um die Zukunftsfähigkeit und Effizienz dennoch zu wahren, ist es unerlässlich, sich diesen Herausforderungen zu stellen. Zusammengefasst können folgende ökonomische Trends bei öffentlichen Verwaltungen identifiziert werden:

Rationalisierung Aufgrund der Kassenlage der öffentlichen Haushalte herrscht ein großer Druck auf eine zunehmende Rationalisierung von Verwaltungsprozessen. Hierbei setzen öffentliche Einrichtungen immer stärker Strategien der Prozessoptimierung, des Outsourcing, d. h. der Auslagerung von Dienstleistungen an Unternehmen, oder Strategien des E-Government, d. h. einer zunehmenden IT-Unterstützung von Verwaltungsabläufen, ein.

Neue Steuerungsinstrumente Ein weiterer Trend ist die Einführung neuer Steuerungsinstrumente in der öffentlichen Verwaltung. Die Einführung z. B. einer Kosten- und Leistungsrechnung ermöglicht es auf einer detaillierteren Ebene, Bereiche

einer Verwaltungsorganisation zu vergleichen und zu bewerten. Solche Steuerungsinstrumente sind ein zentrales Element der Verwaltungsmodernisierung. Darüber hinaus können die Daten der Kosten- und Leistungsrechnung über ein zielgruppengerechtes Berichtswesen einem Leistungsvergleich zugeführt werden.

Strategische Restrukturierung Innerhalb der öffentlichen Verwaltungen ist die strategische Restrukturierung von Aufgaben und Verantwortlichkeiten, die sich je nach Problemlage in einer Zentralisierung oder Dezentralisierung niederschlägt, ein sehr wichtiger Gestaltungsbereich. Diese beiden Entwicklungsrichtungen erscheinen zunächst gegenläufig, hängen aber von der verfolgten Zielsetzung ab. Die Zentralisierung soll dabei helfen, Doppelarbeiten zu vermeiden und durch eine gebündelte Durchführung von Prozessen Größenvorteile zu erlangen. Die Dezentralisierung von Aufgaben- und Budgetverantwortung hat zum Ziel, bisher unerkannte Potenziale und Leistungsreserven durch die Erhöhung von Wettbewerb zu erschließen und im Zusammenspiel mit den neuen Steuerungsinstrumenten ein ausgeprägteres Kostenbewusstsein zu erzeugen.

Wie in einer weiteren Studie (Gölz und Hofmann 2005) des *Fraunhofer IAO* gezeigt werden konnte, gibt es im kommunalen Bereich praktisch keine Gestaltungsfelder[6] mehr, die ohne Berücksichtigung einer adäquaten IT- Unterstützung angegangen werden können. Die allgemein hohe Durchdringung von Verfahren und Abläufen mit IT legt den Schluss nahe, dass dies nicht nur für kommunale, sondern weitgehend ebenso für andere öffentliche Einrichtungen gilt. Zusammengefasst können im Bereich der Informationstechnik folgende Trends bei öffentlichen Dienstleistern identifiziert werden:

Zunehmender IT-Einsatz Es findet ein immer stärkerer Einsatz von IT in der öffentlichen Verwaltung statt. Hervorgerufen wird diese Entwicklung durch mehrere Faktoren, z. B. aus den ökonomischen Bereichen wie Rationalisierung und Dezentralisierung. Beides ist ohne den Einsatz oder die Unterstützung von IT nicht denkbar. Hinzu kommt als Treiber der verstärkte Wunsch der Bürger und der Wirtschaft, mehr Dienste auf elektronischem Wege, z. B. über das Internet, abwickeln zu können. Diese Entwicklung lässt sich ganz allgemein unter dem Stichwort E-Government zusammenfassen. E-Government soll schnellere und qualitativ bessere Dienstleistungen für die Adressaten bieten.

Mehr Webanwendungen Aus interner Sicht der öffentlichen Verwaltung findet eine kontinuierliche Entwicklung hin zu webbasierten Anwendungen statt (Bundesamt

6 Die in der Studie identifizierten kommunalen Handlungsfelder reichen von Personalservices, IT-Services, Immobilienmanagement, Bürgerservice, Wahlangelegenheiten, Bereitstellung von Services in bestimmten Lebenslagen, Betreuung von Kindern, Räumliche Ordnung, Bauordnung, Versorgung, Abfallentsorgung, Bereitstellung öffentlicher Infrastruktur bis hin zur Wirtschaftsförderung.

für Sicherheit in der Informationstechnik 2003, S. 12 ff.). Die damit verbundene Neuentwicklung oder Ablösung von konventionellen, meist proprietären Anwendungen bietet die Chance für den Einsatz von Open-Source-Software. Die Lebenszyklen von Fachanwendungen werden teilweise gezielt ausgenutzt, um eine Plattformunabhängigkeit von Applikationen durch deren webbasierte Umsetzung zu erreichen. In diesen Zyklus fallen Beschaffungsentscheidungen für neue Benutzer- oder Serverplattformen, die nicht mehr an die Systemanforderungen der proprietären Altsysteme gebunden sind.

Standardisierung Mit der Neuentwicklung und Ablösung von Software geht in der Regel eine zunehmende Standardisierung von einzelnen Anwendungskomponenten einher. So setzen sich z. B. bei der Maschine-zu-Maschine-Kommunikation immer mehr Standards auf Basis von XML[7] und SOAP[8] durch. Die oftmals geschlossenen, monolithischen IT-Fachverfahren werden zukünftig stärker daran gemessen werden, ob deren Entwicklung und Betrieb wirtschaftlich sind und Interoperabilität über Systemgrenzen hinweg gewährleistet ist. Die zunehmende Standardisierung ermöglicht einen einfacheren Austausch von Daten und Komponenten und erlaubt darüber hinaus das Verfolgen so genannter Best-of–Breed-Ansätze.

Die aufgeführten Trends schlagen sich natürlich auch in den Ergebnissen der Studie wieder. Nachfolgend sollen die Ergebnisse im Bereich der öffentlichen Verwaltung dargestellt werden.

Open Source ist in der öffentlichen Verwaltung kein unbekanntes Thema. 79 Prozent der befragten öffentlichen Einrichtungen beschäftigen sich mit verschiedenen Einsatzmöglichkeiten von Open Source. Etwas mehr als die Hälfte setzt sich sogar schon seit längerer Zeit mit dem Thema Open-Source-Software auseinander. Das heißt bei einer großen Gruppe der Befragten sind Erfahrungen und sogar gute Kenntnisse bezüglich des aktuellen Diskussionsstands und des Einsatzes von Open-Source-Software in der öffentlichen Verwaltung vorhanden. Dabei ist der Verbreitungsgrad von Open-Source-Software bei großen Einrichtungen höher als bei kleinen Einrichtungen.

Bei näherer Analyse der individuellen Motive und vor dem Hintergrund der aufgeführten aktuellen technologischen Trends erscheint die hohe Aufmerksamkeit, die dem Thema Open-Source-Software zukommt, wenig überraschend. Diese Entwicklungen bieten naturgemäß die Chance, bisher verwendete Plattformen durch neue, kostengünstigere und offenere Plattformen abzulösen. Oftmals bieten Release-Zyklen und geänderte Wartungsbedingungen einen Anlass dazu. Vorteilhaft für die öffentlichen

7 Extensible Markup Language (engl. für „erweiterbare Auszeichnungssprache"), abgekürzt XML, ist ein Standard zur Erstellung maschinen- und menschenlesbarer Dokumente in Form einer Baumstruktur (Quelle: http://de.wikipedia.org/wiki/XML).

8 SOAP (ursprünglich für Simple Object Access Protocol) ist ein Protokoll, mit dessen Hilfe Daten zwischen Systemen ausgetauscht und entfernte Programmaufrufe durchgeführt werden können (Quelle: http://de.wikipedia.org/wiki/SOAP).

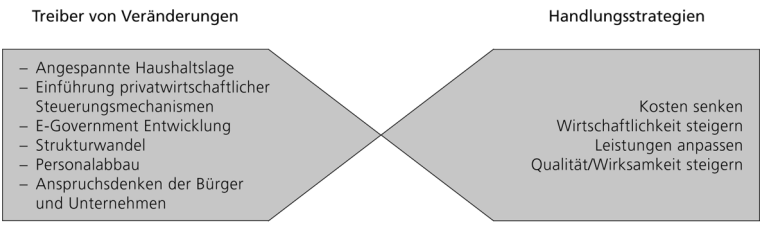

Abbildung 7: Handlungsfelder öffentlicher Verwaltungen (nach Gölz, Hofmann, 2005)

Abbildung 1: Handlungsfelder öffentlicher Verwaltungen (nach Gölz und Hofmann 2005).

Einrichtungen ist dabei, dass bei Open-Source-Software der Quelltext offen liegt, damit also auch prinzipiell Anpassungen durch öffentliche Einrichtungen entweder selbst vorgenommen werden können beziehungsweise die Abhängigkeit von einzelnen (Dienst-)Leistern sinkt, da sie leichter austauschbar werden. Dabei setzen sich die öffentlichen Einrichtungen gezielt und systematisch mit den Möglichkeiten, die der Einsatz von Open Source bietet, auseinander und sind nicht, wie oftmals behauptet, nur punktuell durch die Einzelmotivation von Mitarbeitern getrieben. Für rund 59 Prozent der öffentlichen Verwaltungen ist die Umstellung auf Open-Source–basierte Anwendungen Teil einer mittel- oder langfristigen Gesamtstrategie. Ziel ist es, systematisch betriebskritische Teile der IT-Landschaft auf Open Source zu migrieren.

Die nach Ansicht der Einrichtungen erzielten Vorteile durch den Einsatz von Open-Source-Software liegen dabei vor allem in einer Senkung der Lizenzkosten und der größeren Unabhängigkeit von einzelnen Softwareanbietern. 63 Prozent der befragten Einrichtungen geben an, dass die Einsparung von Lizenzkosten gegenüber kommerziellen Produkten bei Beginn der Open-Source-Projekte im Vordergrund stand. Für 50 Prozent war die größere Unabhängigkeit von Softwareherstellern wichtig. Nicht jeder Release-Wechsel und das damit verbundene Auslaufen von Supportverträgen soll dazu führen, Komponenten austauschen oder vorhandene Software auf eine neue Version bringen zu müssen. Nach Einschätzung der befragten Einrichtungen konnten im Nachblick durch die Einführung von Open Source die anfänglich erwarteten Kostensenkungen bei Lizenzen und die angestrebte, größere Unabhängigkeit von Herstellern auch tatsächlich erreicht werden. Dabei gehen 47 Prozent der Teilnehmer von Kostensenkungen von mehr als 50 Prozent im Bereich der Lizenzkosten aus. Weitere 20 Prozent glauben an eine Kostensenkung von bis zu 25 Prozent. Bei den Betriebskosten und eigenen Personalkosten erwarten die Teilnehmer kaum gravierende Änderungen. Kostensteigerungen werden vor allem bei den Dienstleistungskosten erwartet. Insgesamt gehen die öffentlichen Einrichtungen von Kostensenkungen durch die Verwendung von Open-Source-Software aus.

Bemerkenswert bei der Umstellung auf Open-Source-Software ist, dass nach Aussage der öffentlichen Einrichtungen „sekundäre Ziele“ wie die Erhöhung der Service-

qualität für Bürger oder auch interne Kunden oder die Beschleunigung von Prozessen teilweise erreicht werden konnten, diese Ziele ursprünglich aber von einer großen Mehrheit der Einrichtungen keine angezielten Effekte waren.

Der Einsatz von Open-Source-Software in öffentlichen Einrichtungen kann verschiedenen Aufgaben dienen:

Ergänzung bestehender Anwendungen durch neue, quelloffene Komponenten. Hierbei können bestehende Anwendungen funktionell durch den Einsatz von Open-Source-Komponenten erweitert werden.

Ablösung bestehender Anwendungen durch neue, quelloffene Anwendungen, z. B. durch den Austausch kommerzieller Produkte, und die Übernahme bestehender Funktionalitäten durch Open-Source-Komponenten.

Neueinführung von Open-Source-Anwendungen für neue Zwecke und neue Bereiche. Zuvor war in diesen Bereichen keine IT-Unterstützung vorhanden.

Unter Berücksichtigung dieser Handlungsalternativen, die aus Sicht der Studienteilnehmer bestehen, ist die wichtigste für diese Einrichtungen die Ergänzung der IT durch quelloffene Anwendungen (56 Prozent der Nennungen).[9] Eine Ergänzung bestehender Anwendungen haben die befragten öffentlichen Einrichtungen vor allem in folgenden Bereichen vorgenommen:

- im Bereich von Office-Anwendungen (68 Prozent),
- in Hardware-nahen Bereichen (65 Prozent),
- in Bereich der Desktop-Systeme (60 Prozent).

Die Anwendungslandschaft wurde demnach vor allem im Desktop-Bereich um quelloffene Lösungen ergänzt. Eine wichtige Rolle spielt dabei die Ergänzung der bestehenden Endnutzeranwendungen um quelloffene Alternativen, wie z. B. den Internet-Browser *Firefox*, das Mailprogramm *Thunderbird* oder die Office-Anwendungen *OpenOffice.org*. Unter Hardware-nahen Bereichen wurden im Kontext dieser Befragung quelloffene Optionen für Telefonanlagen oder auf quelloffener Firmware basierende (WLAN-)Router oder Firewalls verstanden.

Die zweitwichtigste Handlungsalternative für die befragten Einrichtungen ist die Ablösung bestehender Anwendungen durch quelloffene Anwendungen (23 Prozent aller Nennungen). Die Ablösung bestehender IT-Anwendungen findet vor allem im Bereich der Server-Betriebssysteme (35 Prozent) und Server-basierten Anwendungen (33 Prozent) statt. Unter Server-basierten Anwendungen in diesem Kontext werden hauptsächlich so genannte Infrastruktur- oder Back-End-Komponenten verstanden, die als Basis für die Entwicklung und den Betrieb von Anwendungen benutzt werden.

9 Die nachfolgenden Zahlen beziehen sich dabei nur auf diejenigen öffentlichen Einrichtungen der Studie, die Open-Source-Software einsetzen.

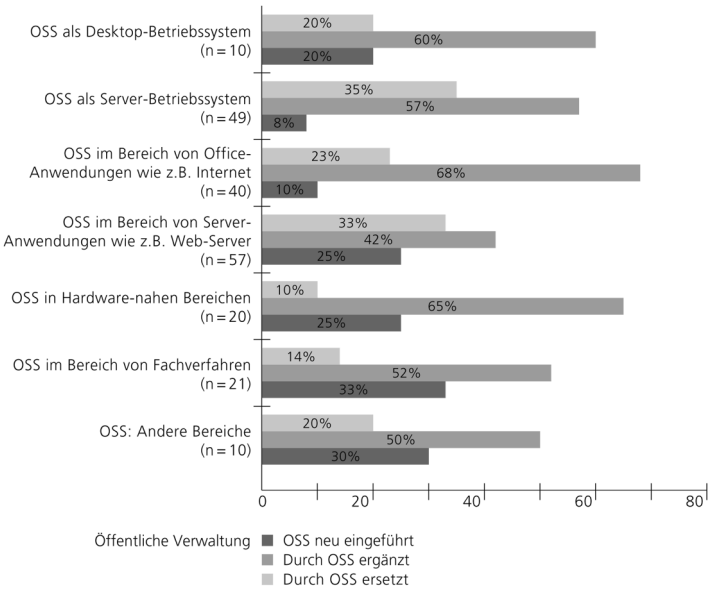

Abbildung 11: Haben Sie in Ihrer Einrichtung nachfolgende informationstechnischen Anwendungen durch OSS ersetzt, ergänzt oder OSS völlig neu eingeführt?

Abbildung 2: Haben Sie in Ihrer Einrichtung nachfolgende IT-Anwendungen durch OSS ersetzt, ergänzt oder OSS völlig neu eingeführt?

Als Komponenten mit dem größten Verbreitungsgrad können hier sicherlich der Webserver *Apache*, die Datenbank *MySQL* oder der Java-basierte *JBOSS Application Server* angesehen werden.

Die Handlungsalternative mit der geringsten Bedeutung für die befragten Einrichtungen ist die *Einführung völlig neuer Anwendungen*, die auf quelloffener Software beruhen (21 Prozent der Nennungen). Dort, wo solche Neueinführungen stattfinden, betrifft es hauptsächlich die Fachverfahren (33 Prozent) und die Server-basierten Anwendungen (25 Prozent).

Dem Bereich der Neueinführung von Fachverfahren kommt OSS-basierter Middleware die wichtigste Bedeutung zu. Hier ist vor allem die Kombination der Bereiche Server-basierte Anwendungen und Fachverfahren interessant, legt sie doch nahe, dass neue Fachverfahren immer stärker auf Basis von Webtechnologien und damit oft auch Open-Source-Technologien umgesetzt werden. Ausgehend von den gegebenen Antworten kann durch die statistische Analyse auch ein Zusammenhang belegt werden.

Ein enger Zeitplan, fehlendes Know-how oder knappe Personalressourcen können Gründe sein, externe Unterstützung in verschiedenen Phasen der Einführung von OSS

hinzuzuziehen. Knapp 25 Prozent der befragten öffentlichen Einrichtungen hatten bei ihren Open-Source-Projekten auf externe Unterstützung zurückgegriffen.

Dabei erhalten vor allem kleine, regionale oder nationale Unternehmen Aufträge von der öffentlichen Hand. Bezogen auf die öffentlichen Einrichtungen, die Aufträge an Dienstleister vergeben haben, stellte diese Gruppe mit einem Anteil von rund 83 Prozent die wichtigsten Dienstleister dar. Mit deutlichem Abstand folgen große, regionale oder nationale Unternehmen (23 Prozent) sowie internationale Unternehmen (17 Prozent).

Der Bereich, in dem am häufigsten auf externe Unterstützung zurückgegriffen wird, ist die eigentliche Projektdurchführung. Unter Projektdurchführung fallen Arbeiten wie Implementierung, Konfiguration und Anpassung von OSS. 73 Prozent der Einrichtungen, die externe Dienstleister einbezogen haben, hatten hierbei externe Unterstützung.

62 Prozent holen sich in der Folge auch Unterstützung beim Support der neuen Anwendungen. Ebenfalls von hoher Bedeutung ist die Unterstützung in der Einführungsphase (61 Prozent). Darunter fallen Arbeiten z. B. bei der Installation und Konfiguration neuer Anwendungen an Arbeitsplatzrechnern oder der Planung und Vorbereitung der dazu notwendigen Vorgehensweise an. Noch knapp die Hälfte beauftragt Dienstleistungen, die mit der Konzeption zusammenhängen, z. B. für die Erstellung von Fach- und DV-Konzepten.

3 Auswirkungen von Open-Source-Software für IT-Unternehmen

Die zunehmende Verbreitung von Open-Source-Software entfaltet seine Wirkungen aber nicht nur wie dargestellt im Anwenderbereich, sondern auch im Bereich der IT-Anbieterunternehmen. In der Studie wurden daher ebenfalls die Auswirkungen, die der zunehmende Einsatz von Open-Source-Software auf die IT-Anbieterunternehmen entfaltet, untersucht.

Im Bereich der IT-Unternehmen konnten insgesamt 94 gültige Antworten erzielt werden. Die Aufforderung zur Teilnahme erging schriftlich an 5000 Unternehmen. Die Ergebnisse der Befragung unter den IT-Anbietern stammen hauptsächlich aus kleineren und mittleren Unternehmen mit bis zu 49 Mitarbeitern. 51 Prozent der Unternehmen, die teilgenommen haben, sind Kleinstunternehmen mit maximal neun Beschäftigten.

Die Aufteilung der Unternehmen auf die Größenklassen entspricht überwiegend den tatsächlichen Verhältnissen, da die meisten IT-Unternehmen in Deutschland kleine und mittelständische Unternehmen sind (Statistisches Bundesamt 2007, S. 6).

Für rund 55 Prozent der befragten IT-Unternehmen spielt Open Source eine mittlere bis sehr große Rolle und macht mindestens 25 Prozent Umsatzanteil aus. Bei 18 Prozent der Unternehmen besteht der Umsatz nahezu vollständig aus OSS–

Dienstleistungen und -Produkten. Dennoch muss festgehalten werden, dass für nur wenige Unternehmen Open Source das ausschließliche Standbein darstellt. Immerhin 35 Prozent der befragten Unternehmen glauben dennoch, dass sie ohne ihr Open–Source-Angebot nicht mehr überlebensfähig wären.

Die Untersuchung der Kundenfelder aller befragten IT-Unternehmen zeigt, dass die Branchen „sonstige Dienstleistungen“[10] und „Unternehmen der großen, produzierenden Industrie“ die größte Bedeutung für die IT-Unternehmen haben. Zwischen OSS- und Nicht-OSS-Unternehmen bestehen allerdings Unterschiede in folgenden Kundenfeldern:

– Öffentliche Verwaltung / Sozialversicherung / Verteidigung
– Bildung und Gesundheit

Beide Bereiche haben für OSS-Unternehmen eine größere Bedeutung als für Nicht–OSS-Unternehmen. So ist der Bereich der öffentlichen Verwaltung mit 59 Prozent sehr wichtig bis durchaus wichtig gegenüber lediglich 44 Prozent bei Nicht-OSS-Unternehmen. Im Bereich Bildung und Gesundheit ist der Unterschied mit 55 Prozent gegenüber 29 Prozent bei Unternehmen, die kein Open Source anbieten, noch bedeutender. Der gesamte Bereich der öffentlichen Verwaltung sowie das Bildungs- und Gesundheitswesen sind seit langen Jahren durch knappe Haushaltsmittel gekennzeichnet. Mögliche Einsparungen durch den Einsatz von OSS im Bereich der IT erscheinen daher in erster Linie für diese Kundenfelder interessant und machen sie gerade daher für OSS-Unternehmen besonders attraktiv. Das Geschäftsmodell von OSS-Unternehmen kommt gerade in diesem Kundenfeld besonders zum Tragen, da der Einsatz von OSS beim Anbieten von Dienstleistungen Kostenvorteile verspricht.

Bemerkenswert ist dabei, wie die Unternehmen OSS innerhalb ihres Produkt- und Leistungsspektrums einsetzen. Für 46 Prozent der Unternehmen stellen Dienstleistungen in Verbindung mit Open Source wie die Implementierung, Anpassung und Einführung von Open Source, beispielsweise Frameworks wie *Typo3* oder *JBOSS*, eine wichtige Komponente in ihrem Produkt- und Dienstleistungsspektrum dar. Weitere 46 Prozent der Unternehmen haben z. B. auf Basis von OSS-Technologien wie *Linux*, *Apache*, *MySQL* und *PHP*[11] eigene Anwendungen entwickelt, die selbst nicht quelloffen sind. Für einen kleineren Teil der IT-Unternehmen, die OSS einsetzen, besteht die Bedeutung von Open Source lediglich darin, dass es eine alternative Betriebsumgebung für eigene, selbstentwickelte, kommerzielle Produkte darstellt. Lediglich 14 Prozent der befragten Unternehmen erstellen Software, die sie ihrerseits wiederum als Open Source anderen Unternehmen frei zur Verfügung stellen.

Wichtigste Triebfedern für die Beschäftigung mit OSS sind die freie Verfügbarkeit sowie das persönliche Engagement beziehungsweise Interesse von Mitarbeitern in

10 *Sonstige Dienstleistungen* umfassen Branchen wie Handel, Banken, Verkehr oder Gastgewerbe.
11 *Linux*, *Apache*, *MySQL* und *PHP* werden auch als sog. LAMP-Systeme bezeichnet.

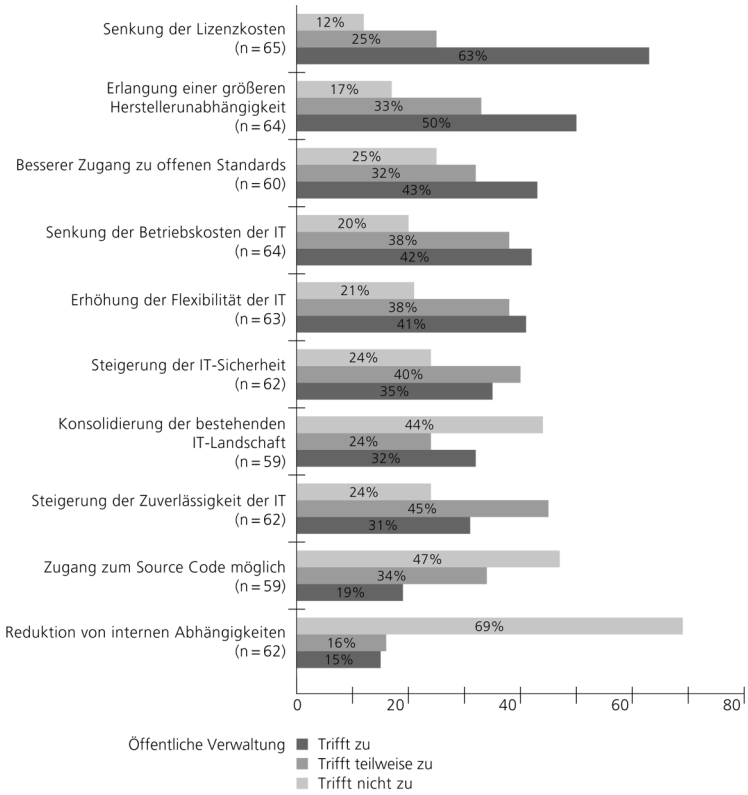

Abbildung 13: Was war die konkrete, messbare Zielsetzung für die Umstellung auf Open Source?

Abbildung 3: Was war die konkrete, messbare Zielsetzung für die Umstellung auf OSS?

diesem Thema. Das sind Motive, die vor allem aus den Unternehmen selbst heraus getrieben und weniger durch externe Faktoren beeinflusst sind. Das wird auch deutlich, wenn die Unternehmen nach ihren Zielsetzungen befragt werden: Je 58 Prozent der befragten IT-Unternehmen schätzen, dass ihnen OSS dabei hilft, ihre eigene Arbeit effizienter zu gestalten und ihnen eine schnellere Nutzung technologischer Weiterentwicklung der Open-Source-Software ermöglicht. Beides liegt sicherlich in dem größeren Grad an Arbeitsteilung begründet, den der Einsatz von Open Source IT-Unternehmen ermöglicht. Die (Weiter-)Entwicklung muss nicht ausschließlich im Unternehmen, sondern kann in den Open-Source-Communitys erfolgen. Nicht jede Funktion oder jeder Fehler muss von Entwicklern des eigenen Unternehmens behoben werden. Teile dieser Aufgaben können an die Open-Source-Community, in der typischerweise solche Open-Source-Software-Projekte angesiedelt sind, zurück-

delegiert werden. Durch diese Arbeitsteilung profitiert jeder in der Community vom Wissen und Know-how der anderen und bietet so vor allem für kleinere Unternehmen Zugang zu Effizienzgewinnen.

Externe Faktoren spielen bei der Beschäftigung mit OSS nur eine untergeordnete Rolle: Dass auch Wettbewerberunternehmen zunehmend Open Source einsetzen oder anbieten, ist nur für einen geringen Anteil (26 Prozent) der befragten Unternehmen ein Anlass, sich mit OSS auseinanderzusetzen. Nur für eine Minderheit (44 Prozent) ist Open Source deshalb interessant, da Kunden explizit danach verlangen.

Dienstleistungen werden von OSS-Unternehmen typischerweise rund um die drei Themenbereiche „Programmierung, Konfiguration oder Anpassung" von OSS an kundenspezifische Bedürfnisse, die Unterstützung bei Rollout beziehungsweise der Einführungsphase sowie Support angeboten. Near- oder Off-Shoring von zu erbringenden Dienstleistungen spielen dabei in der internen Leistungsgestaltung nur eine untergeordnete Rolle. Die Wertschöpfung verbleibt also tatsächlich im größten Anteil der Fälle auch regional beziehungsweise national.

Für die kommenden Jahre schätzen die OSS-Unternehmen, dass Open-Source–Software in Anwenderunternehmen generell an Bedeutung zunehmen wird. Auch für Nicht-OSS-Unternehmen wird es immer wichtiger, sich damit auseinanderzusetzen. Insofern stellt sich die Frage, wie lange noch große Teile der bisher *traditionell* ausgerichteten Softwareunternehmen ohne zumindest ergänzende OSS-Angebote weiterhin marktfähig sein werden können. Bedeutung erlangt das Thema Open Source aus Sicht der IT-Unternehmen vor allem auch immer dann, wenn Open-Source-Komponenten in Standardprodukten, die sie bereits anbieten, zum Einsatz kommen.

Rund 71 Prozent der IT-Unternehmen schätzen, dass der Einsatz von OSS bei ihnen zu verwertbaren Produktinnovationen führt. Das Innovationspotenzial von Open-Source-Software ist begründet in den spezifischen Eigenschaften der OSS. Dabei kommen die wesentlichen Merkmale von Open-Source-Software zum Tragen: die Offenheit des Quellcodes und eine freie Lizenz, die eine fast uneingeschränkte Nutzung der Software zulassen. Verglichen mit proprietärer Software eröffnen diese Eigenschaften Unternehmen ein besonderes Potenzial für Innovationen. Die Ursachen für die hohen Innovationspotenziale in Open-Source-Unternehmen können in folgenden Punkten gesehen werden (Mundhenke 2005, S. 42 f.):

Schnelle Diffusion von Wissen Die Zusammenarbeit von Entwicklern in Open-Source–Communitys unter Ausnutzung von Internettechnologien ermöglicht die schnelle Verbreitung des in der Open-Source-Software enthaltenen Wissens oder angewandter Vorgehensweisen.

Keine Exklusivität von Wissen Die öffentliche Zugänglichkeit zum Quellcode ermöglicht es auch kleinen Unternehmen und Freelancern, vom Wissen anderer zu profitieren.

Komplementäre Angebote Durch die Nutzung von offenen Standards wird es für Dritte einfacher, mögliche ergänzende Programme oder Programmbestandteile zu erstellen.

4 Zusammenfassung

Als wesentliche Ergebnisse der Studie können folgende Punkte festgehalten werden:

- Im Bereich der Lizenz- oder Betriebskosten führt Open-Source-Software nach Einschätzung der befragten öffentlichen Einrichtungen zu teilweise erheblichen Einsparungen.
- Der Einsatz von OSS in öffentlichen Einrichtungen steigert die regionale und nationale Wertschöpfung. Überwiegend regionale und nationale IT-Unternehmen werden von öffentlichen Einrichtungen beauftragt und fördern dadurch den Erhalt oder die Schaffung von Arbeitsplätzen in Deutschland.
- OSS-Unternehmen sind innovativ: Nach Aussagen der IT-Unternehmen führt Open Source in ihren Unternehmen direkt zu mehr Produktinnovationen.
- OSS spielt bereits heute für IT-Unternehmen eine bedeutende Rolle und wird nach Einschätzung der Unternehmen zukünftig noch an Bedeutung gewinnen.

Das *IAO* konnte im Rahmen seiner Studie feststellen, dass vor allen Dingen mittelständische IT-Unternehmen von den Investitionen des öffentlichen Sektors im Zusammenhang mit Open-Source-Software profitieren. Rund 83 Prozent der Aufträge der öffentlichen Verwaltung in diesem Zusammenhang gehen an kleine und mittlere, regionale Dienstleister. Damit trägt Open-Source-Software direkt zu mehr regionaler Wertschöpfung bei. Der Einsatz von Open-Source-Software kann daher nicht nur der öffentlichen Hand Vorteile bringen, sondern auch zu mehr Beschäftigung führen. Eine Studie, die von der EU-Kommission in Auftrag gegeben und kürzlich veröffentlicht wurde, kam dabei zu einem ähnlichen Ergebnis (Ghosh et al. 2006). Dabei kann Europa als die führende Region im Beitrag und der Entwicklung von Open-Source-Software angesehen werden. Nach Aussage der EU-Studie würde eine Stärkung der Weiterentwicklung von Open-Source-Software helfen, die Investitionslücke zu den USA zu schließen. Dabei gehen die Autoren der Studie davon aus, dass Dienstleistungen in Verbindung mit Open-Source-Software bis zum Jahre 2010 rund 32 Prozent des Gesamtanteils an Dienstleistungen ausmachen werden und vor allem auch kleinen und mittleren europäischen Unternehmen durch neue Geschäftsmodelle neue Möglichkeiten eröffnen werden.

Der hohe Anteil von Open Source bei der Neueinführung von Fachverfahren erhöht damit auch den Druck auf Anbieter proprietärer Verfahren, sich mit Open Source auseinanderzusetzen, um auch für die Zukunft ihre Wettbewerbsfähigkeit zu erhalten. Um regionale Beschäftigung zu fördern und zukünftige Innovationen auf

Basis des „öffentlichen Guts“ Open-Source-Software zu unterstützen, sollte eine stärkere staatliche Förderung von Projekten im Umfeld von Open-Source-Software erfolgen.

Öffentliche Einrichtungen sollten auf Grund der aufgeführten Vorteile aber nicht blindlings mit der Einführung von Open-Source-Software beginnen. Zunächst sollte für die umzustellenden Anwendungsbereiche eine genaue Betrachtung der so genannten *Total Cost of Ownership* (*TCO*) erfolgen, um alle mit einer Migration oder Ablösung verbundenen Folgen in einer Kostenaufstellung zu vergleichen und die Vorteilhaftigkeit genau zu prüfen. Zusätzlich kann es im Einzelfall gerade in größeren Einrichtungen notwendig sein, aus organisatorischer Sicht beziehungsweise aus Sicht des IT-Managements Richtlinien festzulegen, die den Einsatz und Umgang mit Open-Source-Software verbindlich festlegen.

Literatur

Bundesamt für Sicherheit in der Informationstechnik (Hrsg.) (2003), *Kommunikations- und Informationstechnik 2010+3: Neue Trends und Entwicklungen in Technologien*, SecuMedia, Ingelheim.

Ghosh, R. A. et al. (2006), Economic impact of open source software on innovation and the competitiveness of the Information and Communication Technologies (ICT) sector in the EU, Study, UNU-MERIT.
http://ec.europa.eu/enterprise/ict/policy/doc/2006-11-20-flossimpact.pdf
[13. Feb. 2008]].

Gosh, R., Krieger, B., Glott, R. und Robles, G. (2002), FLOSS – Free/Libre and Open Source Software: Part IIB, Survey and Study commissioned by the EU, International Institute of Infonomics, University of Maastricht und Berlecon Research GmbH.
http://www.infonomics.nl/FLOSS/report/FLOSSFinal_2b.pdf [16. Jan. 2008].

Gölz, A. und Hofmann, J. (2005), Verwaltung im Umbruch – Strategien zur Verwaltungsmodernisierung, Studie, Fraunhofer IAO.

Mundhenke, J. (2005), Open Source im Wettbewerb – Gesamtwirtschaftliche Potenziale von OSS und Ansätze zur Gewährleistung von Chancengleichheit bei der Softwareentwicklung, Arbeitspapier, Institut für Weltwirtschaft, Kiel.

Statistisches Bundesamt (2007), *Strukturerhebung im Dienstleistungsbereich – Datenverarbeitung und Datenbanken*, Statistisches Bundesamt, Wiesbaden.

Messung von Offenheit an IT-Artefakten
Der Information Technology Openness Benchmark

JAN SUHR

Große Organisationen sind heutzutage häufig mit Altsystemen konfrontiert, die nur aufwendig durch neue Systeme zu ersetzen sind. Offene Architekturen sind ein weit verbreiteter Ansatz, um dieses „Legacy-Problem" zu lösen und um nachhaltige IT-Landschaften zu erzielen. Leider kann der Grad an Offenheit nur schwer ermittelt und verglichen werden, insbesondere auf Entscheidungsebenen auf denen möglicherweise kein ausreichendes Fachwissen vorherrscht. Um die Bewertung der Offenheit von IT-Artefakten – also Systemen, Protokollen und Formaten – zu ermöglichen, wird daher der *Information Technology Openness Benchmark* (ITOB) vorgestellt. Der ITOB kombiniert eine entsprechende Bewertungsmethode mit Kriterien, die offene Computerartefakte charakterisieren. Dies erlaubt es, IT-Artefakte auf einfache Weise zu evaluieren und zu bewerten. Dadurch können diese miteinander verglichen und Schwachstellen identifiziert werden. Um eine einfache Nutzung zu ermöglichen, steht zusätzlich eine Web-basierte Implementierung zur Verfügung.

Schlüsselwörter: Offenheit · Messung · offene Standards · offene Formate · Open Source Software · FLOSS · Altsysteme · IT

1 Eine kurze Geschichte offener Systeme

In den letzten Jahren wurde in der IT die Bezeichnung „offen" bzw. „open" zunehmend populär. Obwohl diese Bezeichnung bereits seit langem verwendet wird, scheint es in jüngster Zeit eine Art Schlagwort geworden zu sein. Open Source (Ensmenger 2004), offene Standards und Open Access[1] sind einige weit verbreitete Bezeichnungen. Darüber hinaus schmücken sich mittlerweile zahlreiche Produkte, Firmen und

1 Siehe http://www.open-access.net [23. Nov. 2007].

Internet-Dienste mit dieser Bezeichnung im Namen. Möglich ist dies, da keine einheitliche Definition von Offenheit in der IT existiert (Gray 1993). Um zu ermitteln was Offenheit im Detail bedeutet, wird daher im Folgenden die Entstehungsgeschichte so genannter offener Systeme kurz geschildert.

Vor den 1980ern waren IT-Landschaften durch zentrale Großrechner, Mini- und Mikrocomputer großer Hersteller geprägt, wie z. B. *IBM*, *DEC* und *Sun Microsystems* (Gray 1993; Wheeler 1993). Die Computerarchitekturen dieser Zeit waren stark zentralisiert und Mitarbeiter griffen auf die *mainframes* über Terminals zu, die selber kaum Funktionalität aufwiesen. Firmen waren in einer ähnlich stark zentralisierten und unflexiblen Art und Weise organisiert (Tapscott und Caston 1993). Die meisten Hersteller verkauften ihre eigene, proprietäre Computerarchitektur zusammen mit zugehöriger Software. Diese Software lief nur auf einer einzigen Architektur und nicht auf Computern anderer Hersteller (Gray 1993; Wheeler 1993). In einigen Fällen war es sogar nicht möglich, Software auf unterschiedlichen Architekturen des gleichen Herstellers zu verwenden. Der Grund hierfür war, dass die Portierung einer Software sehr aufwendig und teuer war, und daher möglichst vermieden wurde (Tapscott und Caston 1993). Ebenso war der Datenaustausch aufgrund proprietärer Netzwerke, Protokolle und Datenformate nur eingeschränkt möglich (Wheeler 1993). Technische Abhängigkeiten sorgten dafür, dass die Änderung einer einzelnen Komponente, z. B. der Software, mitunter die Änderung einer anderen Komponente, z. B. der Hardware, nach sich zog. Zusätzlich kamen mit der Einführung neuer Systeme die Kosten für Schulungen hinzu, da sich die Benutzerschnittstellen mitunter stark unterschieden. Aus diesen Gründen verursachten Systemwechsel sehr hohe Kosten. Hatte man sich für einen Systemlieferanten entschieden, war man daher stark von diesem abhängig (Varian und Shapiro 2007). Folglich hatten große Hersteller kein Interesse an offenen und herstellerneutralen Standards, die eine bessere Interoperabilität und einen intensiveren Wettbewerb zur Folge gehabt hätten (Tapscott und Caston 1993). Als Meilenstein gilt daher der *American Standard Code for Information Interchange*, der es erstmals ermöglichte, einfache Textdateien zwischen Systemen unterschiedlicher Hersteller auszutauschen. ASCII wurde Ende der 50er Jahre entwickelt, aber es dauerte 15 Jahre, bis der Standard sich weitgehend verbreitet hatte (Freyermuth 2007).

Beginnend in den 70ern bis in die 90er Jahre wurden, insbesondere in der geschäftlichen IT, die so genannten *offenen Systeme* zu einem neuen Paradigma.[2] Dieses zeichnete sich primär durch plattformunabhängige Software, einheitliche Benutzerschnittstellen, herstellerunabhängige Protokolle und Datenformate aus und erlaubte daher den Datenaustausch zwischen Systemen unterschiedlicher Hersteller. Da das Betriebssystem *UNIX*[3] viele dieser Anforderungen realisierte, wurde es häufig – aber

2 Der genaue zeitliche Höhepunkt dieser Entwicklung unterscheidet sich in der Literatur.

3 Ende der 60er und Anfang der 70er entwickelte Ken Thompson, damals Forscher bei den Bell Labs, das Betriebssystem *UNICS*, welches später *UNIX* genannt wurde. Es zeichnete sich durch Quelloffenheit und Plattformunabhängigkeit aus, was zu einer starken Verbreitung beitrug.

fälschlicherweise – als die Manifestation offener Systeme angesehen. Stattdessen bedeuten offene Systeme einen weiterreichenden Ansatz, der unter anderem auch die Hardware betrifft (Bues 1995). In dieser Ära wurde der Personal Computer (PC) erfunden und verbreitete sich zunehmend. Der PC stellte zu dieser Zeit eine neue, leistungsfähige und kostengünstige Alternative dar. Deshalb wurden PCs immer mehr für Aufgaben verwendet, die zuvor teuren *mainframes* oder Minicomputern vorbehalten waren. Dies erlaubte es, die bisherige zentrale Architektur durch eine neue Client/Server-Architektur zu ersetzen. Damit wurde es auch möglich, Daten mittels eines Netzwerks direkt zwischen PCs auszutauschen, was dem Konzept offener Systeme entsprach. Dies gab Firmen die Flexibilität, sich modularer und dynamischer zu organisieren. Die reduzierte Herstellerabhängigkeit resultierte in einem geringeren Risiko für die Anwender und in einem intensiveren Wettbewerb. Es wurde möglich, einen Hersteller zu wechseln, ohne die gesamte IT ersetzen zu müssen. Unabhängige Softwareproduzenten erlaubten es Benutzern, zwischen unterschiedlichen Anbietern und ihren plattformunabhängigen Produkten zu wählen. Die Folge war, dass Hersteller günstigere Software entwickeln konnten, da diese mit geringerem Aufwand portiert werden konnte. Darüber hinaus wurden die verwendeten Benutzerschnittstellen mehr und mehr einheitlich. Dies verringerte die Kosten, Mitarbeiter in der Benutzung von Computern zu schulen. Aus den genannten Gründen sparten Organisationen, die IT verwendeten, Kosten und erhielten gleichzeitig höherwertige Produkte (vgl. Tapscott und Caston 1993; Wheeler 1993; Gray 1993).

Heute werden IT-Landschaften durch PCs dominiert, die zu einem großen Teil auf *Intels* x86-Architektur basieren, zu der die meiste Software kompatibel ist. Software von *Microsoft* hat auf Desktop-PCs eine dominierende Marktstellung, allerdings gibt es viele herstellerunabhängige Protokolle und Dateiformate, die einen plattformunabhängigen Datenaustausch ermöglichen. Anders als auf dem Desktop sind die Softwarelandschaften bei Servern heterogener und häufig stammt die Software von unterschiedlichen Herstellern. Hier werden auch die meisten *UNIX*-Systeme verwendet (Cooper 2004).

In der letzten Dekade, mit dem Erfolg des Internets und des World Wide Webs (WWW), entstand eine neue Generation plattformunabhängiger Systeme. Maßgebliche Voraussetzung hierfür waren plattformunabhängige Protokolle und Formate, die es ermöglichten, Computer weltweit miteinander zu vernetzen. Diese verbesserte Interoperabilität stellt einen Schlüssel für den Erfolg des Internets dar (Cooper 2004). Mittlerweile sind viele Anwendungen über das WWW verfügbar, die zuvor nur als lokale Programme existierten. Im Idealfall ist jeder Webbrowser unabhängig von seiner Plattform in der Lage, auf jede beliebige Webanwendung zuzugreifen.

Zusammenfassend lässt sich sagen, dass sich die Offenheit von IT-Systemen und damit die Herstellerunabhängigkeit im Laufe der Zeit verbessert hat. Nach wie vor existiert aufgrund technischer Merkmale eine teilweise bedeutende Herstellerabhängigkeit. Daher ist es nicht verwunderlich, dass die Kosten einer Systemumstellung

häufig die gesamten Anschaffungskosten von Hardware und Software übersteigen (vgl. Varian und Shapiro 2007; Lessig 1999).

2 Der IT Openness Benchmark

Offene Computerartefakte sind ein viel versprechender Ansatz, um flexible und nachhaltige IT-Landschaften zu erzielen. Ein grundlegendes Problem bei der Durchsetzung der Offenheit ist allerdings dessen unklare Definition. Daher lässt sich der Grad an Offenheit vieler Artefakte nicht leicht erkennen. Diesem liegt das so genannte Saure-Gurken-Problem (engl. *market of lemons*) zugrunde, welches eine asymmetrische Informationsverteilung zwischen einem Verkäufer und einem (potentiellen) Käufer beschreibt. Anders als der Käufer kennt der Verkäufer die wirkliche Qualität des angebotenen Produktes. Der Käufer kann die Qualität der Ware nur eingeschränkt beurteilen, ist sich aber über seine Unwissenheit im Klaren. Um keinen möglicherweise überteuerten Preis zu zahlen, wählt er eine günstigere Alternative, so dass der Markt zu minderwertigen und günstigen Produkten tendiert. Ist hingegen die Qualität ausreichend erkennbar, ist der Käufer möglicherweise bereit den Mehrwert[4], zu bezahlen, so dass der Markt zu höherwertigen Produkten tendieren kann.

Der Zweck des *Information Technology Openness Benchmark* (ITOB)[5] ist es daher, die Offenheit von IT-Artefakten messbar zu machen. Dadurch wird der Mehrwert der Offenheit erkennbar und vergleichbar. IT-Artefakte bezeichnen hierbei Software- und Hardware-Systeme aber auch Protokolle und Formate. Der ITOB besteht aus einer Bewertungsmethode und aus Kriterien, die offene Computerartefakte charakterisieren. Da keine adäquate Definition von Offenheit existiert, wurden Kriterien identifiziert, die Offenheit im Sinne von Benutzerunabhängigkeit und -flexibilität wiederspiegeln (vgl. Open ePolicy Group 2005). Darüber hinaus sind diese Kriterien generisch genug, um auf unterschiedliche Artefakte anwendbar zu sein.

2.1 Kriterien

Im Folgenden werden die zentralen Kriterien des ITOB kurz dargestellt, und auf einer 4-Punkte-Skala definiert. Dabei wird nach Systemen (Software und Hardware) und Spezifikationen (Protokolle und Formate) unterschieden, und diese Aufteilung entsprechend in Tabellen gegliedert.

4 Gemeint ist der eigentliche Sinne des Wortes.

5 Siehe Suhr (2007) für Details.

6 In Anlehnung an einen wohlwollenden Diktator bezeichnet ein egoistischer Diktator eine Person, die ein Sponsoring kontrolliert, hierbei aber nur nachrangig die Interessen der Anwender und Unterstützer berücksichtigt.

7 Ein wohlwollender Diktator bezeichnet eine Person, die ein Sponsoring im Interesse der Anwender und Unterstützer kontrolliert bzw. koordiniert.

8 Engl. *reasonable and non discriminatory*.

Bewertung	Sponsoring von Systemen	Sponsoring von Spezifikationen
0 – ungenügend	Keine Beeinflussung vorgesehen; Sponsoring durch eine einzige Instanz oder monokratische Community	Sponsoring durch eine einzige Instanz oder monokratische Community; keine Beeinflussung vorgesehen; unveröffentlichte und proprietäre Spezifikation
1 – eingeschränkt	Unkooperativ oder nur eingeschränkt kooperativ; egoistischer Diktator[6]; monokratische oder oligarchische Community	Mehrstimmiger Entscheidungsprozess, der von einer einzigen Organisation kontrolliert wird. Proprietäre Spezifikation, die aber weitreichend akzeptiert ist, so dass es faktisch nicht mehr durch eine einzige Institution kontrolliert wird.
2 – befriedigend	Kooperative und informelle Organisation oder wohlwollender Diktator[7]	Offener Einigungsprozess; offener Standardisierungsprozess; keine rechtmäßige Standardisierungsorganisation
3 – gut	Auf Kooperation und Beteiligung basierte demokratische Stiftung oder Verein	Spezifikation wird durch eine offene oder rechtmäßige Standardisierungsorganisation gesponsert.

Tabelle 1: Sponsoring auf einer 4-Punkte-Skala definiert.

Bewertung	Kompatibiltät von Systemen	Kompatibilität von Spezifikationen (bzgl. Implementierungen)
0 – ungenügend	Eingeschränkt; *Digital Rights Management* (*DRM*); verschlüsselt	Eingeschränkt; *Digital Rights Management* (*DRM*); verschlüsselt
1 – eingeschränkt	Vollständig kompatibel zu einer oder wenigen Implementierungen	Proprietäre Spezifikation; wird von einer oder wenigen Implementierungen verwendet
2 – befriedigend	Vollständig kompatibel mit mittelmäßig vielen Implementierungen	Begrenzte Verbreitung
3 – gut	Entwickelt unter Beachtung der Kompatibilität; kompatibel mit den meisten Implementierungen	Weite Verbreitung; entwickelt unter Beachtung der Kompatibilität; dominierende Marktverbreitung

Tabelle 2: Kompatibilität auf einer 4-Punkte-Skala definiert.

Sponsoring bezeichnet die Pflege und Weiterentwicklung von IT-Artefakten, aber auch die Unterstützung von Benutzern. Es bezieht sich auf den Umfang und die Art und Weise der Unterstützung, sowie die Person oder Organisation die dies leistet. Computerartefakte werden in der Regel durch einzelne Personen, Firmen, lose Communitys, Vereine, Stiftungen oder Standardisierungsgremien gesponsert. Für Externe variieren diese Organisationsformen in der Möglichkeit, das Sponsoring zu unterstützen bzw. zu beeinflussen. Je offener ein Sponsoring ist, desto mehr sind Benutzer in der Lage, die Entwicklung des Artefakts bzgl. eigener Anforderungen zu beeinflussen. Dies ist insbesondere für Standards

Bewertung	Interoperabilität von Systemen	Interoperabilität von Spezifikationen (bzgl. Implementierungen)
0 – ungenügend	Eingeschränkt; *Digital Rights Management* (*DRM*); verschlüsselt	Eingeschränkt; *Digital Rights Management* (*DRM*); verschlüsselt
1 – eingeschränkt	Vollständig kompatibel zu einer oder wenigen Implementierungen	Proprietär; entwickelt und verfügbar nur für eine einzige Organisation
2 – befriedigend	Interoperabilität ist mittelmäßig ausgeprägt	Mittelmäßiger Marktanteil
3 – gut	Gute Interoperabilität; entwickelt unter Beachtung der Interoperabilität	Weite Verbreitung; dominierender Marktanteil

Tabelle 3: Interoperabilität auf einer 4-Punkte-Skala definiert.

Bewertung	Verfügbarkeit von Systemen	Verfügbarkeit von Spezifikationen
0 – ungenügend	Proprietär; Quelltext steht nicht allen Nutzern zur Verfügung	nicht verfügbar
1 – eingeschränkt	Proprietär, aber Quellcode steht allen Nutzern zur Verfügung – Vollständig kompatibel zu einer oder wenigen Implementierungen	Teilweise verfügbar (z. B. durch *reengineering*)
2 – befriedigend	Bewahrende Open-Source-Lizenz	Vollständig verfügbar, nicht standardisiert
3 – gut	Freizügige Open-Source-Lizenz	Vollständig verfügbar und standardisiert

Tabelle 4: Verfügbarkeit auf einer 4-Punkte-Skala definiert.

wichtig, weil deren Offenheit nicht über inhärente technische Eigenschaften, sondern über den zu Grunde liegenden Standardisierungsprozess definiert wird (vgl. Merten und Meretz 2005; siehe Tabelle 1). Zusätzlich sollte ein gewissenhaftes Sponsoring sicherstellen, dass ein Artefakt heute und auch in der Zukunft umfassend unterstützt wird. Daher ist ein Sponsoring durch unterschiedliche und eine Vielzahl von Personen bzw. Institutionen vorteilhaft.

Bewertung	Patente
0 – ungenügend	Definitiv durch Patente betroffen; Nutzung ist gegen eine hohe Gebühr oder gar nicht zu erwerben
1 – eingeschränkt	Unentschieden ob von Patenten betroffen, aber mit hohem Risiko
2 – befriedigend	Unentschieden ob von Patenten betroffen, mit geringem Risiko; Nutzung aufgrund von RAND-Bedingungen[8] möglich
3 – gut	Nicht von Patenten betroffen; kostenlose Nutzung ist unwiderruflich gewährt; Produkte mit geschlossenem Quellcode

Tabelle 5: Patente auf einer 4-Punkte-Skala definiert.

Kompatibilität bezieht sich auf die Fähigkeit, dass unterschiedlicher Computerartefakte in der *gleichen* Umgebung Daten austauschen. Gooch (2003) beschreibt Kompatibilität als die Eigenschaft von Teilen, in einem System zusammen zu arbeiten (siehe Tabelle 2).

Interoperabilität ist die Fähigkeit von zwei oder mehr Systemen in *unterschiedlichen* Umgebungen, über vordefinierte Methoden zu interagieren, um ein vorhersagbares Ergebnis zu erzielen (Brock und Roston 1996). Die Bezeichnung Interoberabilität hat ihren Ursprung in der militärischen Terminologie, wo sie die Fähigkeit der Kooperation von Komponenten in unterschiedlichen Umgebungen beschreibt (Gooch 2003).

Da sich Interoperabilität und Kompatibilität beide auf den Datenaustausch unterschiedlicher Systeme[9] beziehen, können diese leicht miteinander verwechselt werden. Zur Veranschaulichung kann ein Schichtenmodell verwendet werden, indem unterschiedliche Systeme übereinander und ähnliche Systeme nebeneinander angeordnet sind. Kompatibilität bezieht sich auf die Vertikale und kann Systeme beinhalten, die komplementär zueinander sind. Zum Beispiel benötigt eine Softwareanwendung ein Betriebssystem, welches wiederum einen Computer benötigt. Diese Komponenten befinden sich in der gleichen Umgebung und können zusammen wiederum ein eigenes System bilden. Interoperabilität bezieht sich hingegen auf die Horizontale, auf Systeme die Substitute sein können, da sie für ein und denselben Zweck verwendet werden können, z. B. zwei unterschiedliche E-Mail-Programme. Üblicherweise ist die Interoperabilität zwischen zwei gleichen Instanzen – also zwischen Kopien – 100 %.

Bei Formaten und Protokollen ist für die Kompatibilität ausschlaggebend, wie viele Implementierungen zu diesen kompatibel sind. Existieren viele kompatible Implementierungen, die allerdings nur einen geringen Marktanteil aufweisen, kann sich die Kompatibilität in der Praxis schnell als nicht ausreichend (im Sinne von Offenheit bzw. Benutzerunabhängigkeit) herausstellen. Daher wird unter Interoperabilität von Spezifikationen deren Marktanteil verstanden (siehe Tabelle 3).

Verfügbarkeit bezieht sich auf die Art und Weise, wie das Artefakt zugänglich gemacht wird. Bei Hardware wird die Verfügbarkeit der Designdokumente bzw. Spezifikationen betrachtet. Software und Spezifikationen werden meistens unter einer bestimmten Urheberrechtslizenz veröffentlicht. Es hängt von den in der Lizenz gewährten Rechten ab, ob das Artefakt mehr oder weniger offen für die Benutzer ist.

Proprietäre Software gewährt den Nutzern wenige Freiheiten und wird nicht als Quellcode veröffentlicht. Open-Source-Software gewährt mehr Freiheiten,

9 Protokolle und Formate werden letzten Endes in Systemen implementiert.

variiert aber auch in der konkreten Ausgestaltung. Hierbei unterscheiden sich bewahrende (engl. *protective*) und freizügige (engl. *permissive*) Open-Source-Lizenzen voneinander. Bewahrende Open-Source-Lizenzen verlangen, dass Derivate der originalen Software (engl. *fork*) unter die gleiche Lizenz wie das Original gestellt werden. Diese Reziprozität wird auch, in Anlehnung an das englische *copyright*, als *copyleft* bezeichnet. Es verhindert, dass Software, die einmal als Open Source veröffentlicht wurde, später als proprietäre Software genutzt wird (vgl. Free Software Foundation 2007). Die *GNU Public License* (*GPL*)[10] enthält die Copyleft-Eigenschaft und ist vermutlich die meist verwendetste Open-Source-Lizenz. Freizügige Open-Source-Lizenzen schränken hingegen nicht die Lizenz von Derivaten ein. Daher ist es möglich, eine Open–Source-Software (zusätzlich) unter einer proprietären Lizenz zu veröffentlichen, ohne den Quellcode zugänglich machen zu müssen (siehe Tabelle 4).

Patente können die Benutzung von Artefakten, die durch sie betroffen sind, einschränken. Ist ein Artefakt von einem Patent betroffen, kann der Patentinhaber dessen Nutzung untersagen. Die meisten Patentbesitzer erlauben jedoch die Nutzung ihres Patentes gegen eine Gebühr.

Um Standards zu verabschieden, die nicht von Patenten betroffen sind, verlangen viele Standardisierungsorganisationen von ihren Mitgliedern, alle den jeweiligen Standard betreffenden Patente unter einer bestimmten Lizenz frei zu geben. Häufig werden dafür so genannte *reasonable and non discriminatory* (RAND) Lizenzbedingungen verwendet, die den Patentinhaber verpflichten, für die Nutzung lediglich eine angemessene Gebühr zu verlangen und keinen Lizenznehmer zu diskriminieren. Andere Standardisierungsorganisationen verlangen von ihren Mitgliedern, die betroffenen Nutzungsrechte der Patente kostenlos zu gewähren (vgl. W3C 2004).[11] Wenn Patente kostenlos genutzt werden können, sollte sichergestellt sein, dass diese Genehmigung nicht später zurückgezogen werden kann (Gehring und Lutterbeck 2003; siehe Tabelle 5).

2.2 Der Bewertungsprozess

Die Kriterien des letzten Abschnitts werden mit einem entsprechenden Bewertungsprozess kombiniert, um ein einheitliches Vorgehen zu gewährleisten. Für eine präzise Bewertung können einzelne Komponenten des Artefakts separat voneinander bewertet werden. Hierzu wird das zu untersuchende Artefakt theoretisch in eine beliebige Anzahl von Ebenen, bzw. Schichten zergliedert, z. B. Hardware-, Betriebssystem- und Anwendungs-Ebene. Um auf diverse unterschiedliche Artefakte anwendbar zu sein, kann der Anwender die benötigten Schichten frei wählen. Die Kriterien des letzten Kapitels werden auf alle *n* Schichten angewandt, so dass sich eine zweidimensionale

10 Sofern nicht anderweitig angegeben, bezieht sich die Nennung der *GPL* auf Version 2.

11 Siehe http://de.wikipedia.org/wiki/RAND_(Lizenzierung) [23. Nov. 2007].

Matrix als Darstellungsweise eignet (siehe Tabelle 6). Die Durchführung der Bewertung ist in den folgenden fünf Schritten beschrieben:

1. *Vorbereitung:* Die passenden Ebenen (vgl. Gray 1993) werden identifiziert und eine entsprechende Tabelle erstellt, um eine einfache Darstellung zu ermöglichen. Zum Beispiel könnten Protokoll, Datenformat, Anwendung, Betriebssystem und Hardware passende Schichten sein. Der Anwender kann die für den jeweiligen Fall geeigneten Schichten frei definieren. Bei Bedarf können außerdem Schwellwerte für einzelne Kriterien, Ebenen oder für ein einzelnes Kriterium einer einzelnen Schicht definiert werden, die von dem untersuchten Artefakt erfüllt werden müssen.

2. *Bewertung:* Alle Kriterien aller Schichten werden mit 0, 1, 2 oder 3 bewertet. Ein geringer Wert gilt für eine geschlossene Ausprägung und ein großer Wert für Offenheit. Die genauen Beschreibungen der Kriterien des letzten Abschnitts gelten hierfür als Grundlage. Wenn Schwellwerte definiert wurden, werden die Ergebnisse anhand dieser überprüft und der Bewertungsprozess ggf. abgebrochen.

3. *Gewichtung der Kriterien:* Um individuelle Präferenzen zu berücksichtigen, werden einzelne Kriterien entsprechend gewichtet. Dazu wird jedes der fünf Kriterien mit einem frei wählbaren Faktor gewichtet. In Summe müssen diese Faktoren 50 Punkte ergeben, was im Durchschnitt einen Faktor von 10 ergibt. Soll aufgrund individueller Präferenzen ein Kriterium nicht bewertet werden, ist dieses mit einem Faktor von 0 zu gewichten. Die ungewichteten Ergebnisse aus dem vorhergehenden Schritt werden mit dem jeweiligen Faktor multipliziert. Als Ergebnis enthält jede Schicht Bewertungen, deren Summe zwischen 0 und 150 Punkten liegt.[12]

4. *Gewichtung der Schichten:* Jeder der n Schichten wird eine frei wählbare Gewichtung zugewiesen. Im Durchschnitt wird eine Gewichtung von 10 Punkten für jede Schicht vergeben, so dass deren Summe $n * 10$ beträgt. Die Ergebnisse des letzten Schrittes werden mit der jeweiligen Gewichtung der Schicht multipliziert, so dass die sich ergebenden Werte zwischen 0 und $n * 1500$ liegen.[13]

5. *Normalisierung:* Um eine Vergleichbarkeit zwischen unabhängig durchgeführten Bewertungen zu ermöglichen, werden die gewichteten Ergebnisse zum Schluss normalisiert. Dafür werden alle Werte durch $n * 10$ dividiert. Anschließend werden alle Werte aufsummiert, um das gewichtete Gesamtergebnis zu erhalten. Für eine einfache Auswertung können die Ergebnisse zusätzlich in einem Netzdiagramm dargestellt werden (siehe Abbildung 1).

12 Die höchste Bewertung ist 3 und die Gewichtung ist 50, so dass gilt: $3 * 50 = 150$.

13 Da die Anzahl der Ebenen n entspricht, die durchschnittliche Gewichtung der Kriterien 10 beträgt und die maximale Punktzahl 150 ist, gilt: $n * 10 * 150 = n * 1500$.

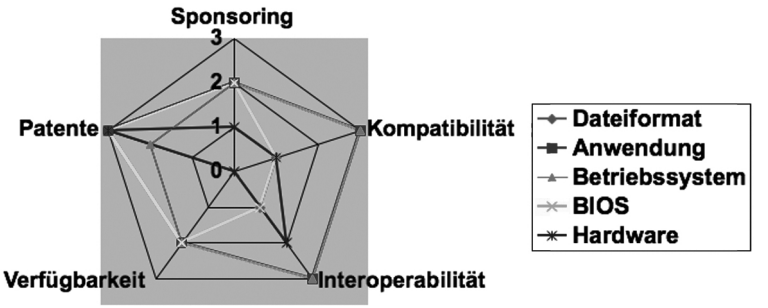

Abbildung 1: Ungewichtete Ergebnisse der Bewertung des XO.

3 Das Beispiel 100-Dollar-Laptop

Das oben beschriebene Vorgehen soll anhand eines realen Beispiels veranschaulicht werden. Hierfür dient der so genannte *100-Dollar-Laptop*, der auch als *One Laptop Per Child* (*OLPC*) oder *XO* bekannt ist. Der Laptop ist mit *GNU/Linux* ausgestattet und dafür ausgelegt, von Schulkindern in weniger entwickelten Ländern verwendet zu werden. Um die Bewertung zu vereinfachen, wird das Gerät ausschließlich in Bezug auf Textverarbeitung untersucht.

Vorbereitend (Schritt 1) werden die Ebenen festgelegt, unter denen das Artefakt betrachtet wird. In diesem Beispiel bieten sich die folgenden Ebenen an:

1. Dateiformat: *AbiWord* speichert Dokumente standardmäßig im eigenen Dateiformat *ABW*.
2. Anwendung: Standardmäßig ist die Textverarbeitung *AbiWord* installiert.
3. Betriebssystem: *GNU/Linux* ist als Betriebssystem vorinstalliert.
4. *BIOS*: *LinuxBIOS* wird zum Starten des Betriebssystems verwendet.
5. Hardware: Die vorliegende Hardware unterscheidet sich zu herkömmlichen Laptops, da es nicht die üblichen Komponenten und Bauformen beinhaltet.

Um den Umfang dieser Darstellung einzuhalten, werden keine Schwellwerte verwendet, die ansonsten in diesem Schritt definiert werden müssten. Im anschließenden 2. Schritt werden die Kriterien jeder Ebene bewertet (siehe Tabelle 6). Als Grundlage hierfür dienten die Beschreibungen der Kriterien in Abschnitt 2.1. Jede Bewertung im Detail zu erläutern würde ebenfalls den Rahmen dieser Darstellung sprengen, so dass auf diese im Folgenden lediglich kurz eingegangen wird. [14]

14 Siehe Suhr (2007) für Details.

Schicht	Sponsoring	Kompatibilität	Interoperabilität	Verfügbarkeit	Patente	Gesamt
Dateiformat	2	1	1	2	3	9
Anwendung	2	3	3	2	3	13
Betriebssystem	2	3	3	2	2	12
BIOS	2	1	1	2	3	9
Hardware	1	1	2	0	3	7
Gesamt	9	9	10	8	14	50

Tabelle 6: Ungewichtete Bewertungen des XO für Textverarbeitung.

Sponsoring *AbiWord* und *ABW* werden durch das so genannte *AbiWord Team* entwickelt, welches eine informelle Community ist, an der insbesondere einzelne Entwickler partizipieren. *GNU!~ Linux* besteht aus dem Linux-Kernel und zahlreicher Software, wie z. B. Bibliotheken, Werkzeugen und einer grafischen Benutzeroberfläche. Diese werden teilweise durch einzelne Entwickler, aber auch durch Firmen und andere Organisationen entwickelt. Vorsichtig betrachtet ergibt sich daher eine Bewertung von zwei Punkten. Die Hardware wird wiederum durch eine Non-Profit-Stiftung entwickelt, die die Unterstützung durch Dritte grundlegend begrüßt. Da nicht eindeutig bekannt ist, unter welchen Umständen man Einblick in das Hardwaredesign nehmen und dessen Entwicklung beeinflussen kann, wird das Sponsoring mit einem Punkt bewertet.

Kompatibilität Das Dateiformat *ABW* ist proprietär, da es ausschließlich von *AbiWord* verwendet wird. *AbiWord* hingegen ist eine sehr kompatible Software, die auf einer Vielzahl von Betriebssystemen läuft. Ebenso läuft *GNU/Linux* auf vielen unterschiedlichen Hardwarearchitekturen. Allerdings kann *LinuxBIOS* nur auf einer eingeschränkten Anzahl von Systemen verwendet werden. Die Hardware des *XO* ist proprietär und nicht zu anderen PCs kompatibel, so dass nur speziell angepasste Betriebssysteme damit funktionieren.

Interoperabilität Wie bereits erwähnt, wird das Format *ABW* ausschließlich von *AbiWord* verwendet. Diese Anwendung weist wiederum einen kleinen Marktanteil an den Textverarbeitungsapplikationen auf. Daher ist die Interoperabilität des Formates *ABW* als gering anzusehen. *AbiWord* kann viele alternative Formate, z. B. *DOC*, *RTF*, *HTML*, *WordPerfect*, nutzen, was eine hohe Interoperabilität auszeichnet. Ebenso kann *GNU/Linux* über viele Formate und Protokolle mit anderen Betriebssystemen interoperieren. Die Hardware verfügt für diesen Zweck über Schnittstellen, wie z. B. *USB*, 802.11b/g kompatibles *WLAN* und einen Speicherkarten-Slot, die aber in der Anzahl und Ausstattung nicht an handelsübliche Laptops heranreichen.

Verfügbarkeit Das Format *ABW* ist vollständig unter der *GPL* veröffentlicht, allerdings

	Sponsoring	Kompatibilität	Interoperabilität	Verfügbarkeit	Patente	Gesamt
Gewichtung	15	10	5	13	7	50

Tabelle 7: Exemplarische Gewichtungen der Kriterien.

Schicht	Gewichtung
Dateiformat	15
Anwendung	10
Betriebssystem	15
BIOS	5
Hardware	5

Tabelle 8: Exemplarische Gewichtungen der Schichten.

nicht standardisiert. *AbiWord*, *GNU/ Linux* und *LinuxBIOS* sind größtenteils unter der *GPL* verfügbar. Dahingegen ist das Hardwaredesign nur einzelnen Entwicklern zugängig und nicht unter einer offenen Lizenz veröffentlicht.

Patente Die Software des *XO* kann als patentfrei angesehen werden, wohingegen bei *GNU/Linux* einige weniger ernstzunehmende Risiken der Patentverletzung bestehen. Da das Hardwaredesign nicht veröffentlicht wurde, besteht kein Risiko durch die Nutzung des Gerätes möglichen Patentansprüchen ausgesetzt zu sein.

Nachdem die Bewertung abgeschlossen ist, sind in Schritt 3 und 4 individuelle Gewichtungen zu vergeben. Zur Demonstration des Berechnungsverfahrens wurden zufällige Gewichtungen gewählt, die daher nicht näher zu begründen sind (siehe Tabelle 7 und 8). Wie im vorherigen Abschnitt beschrieben, sind diese Gewichtungen mit den Ergebnissen des vorherigen Schrittes zu multiplizieren. Um ein normalisiertes Ergebnis zu erhalten, werden diese Werte anschließend im 5. Schritt durch $n * 10$ dividiert, wobei n die Anzahl der Ebenen darstellt. Bei fünf Schichten entspricht dies einem Divisor von 50. Die gewichteten und normalisierten Ergebnisse sind in Tabelle 9 dargestellt.

Die Ergebnisse können zum einen genutzt werden, um Schwachstellen aufzuzeigen, zum anderen können diese in einem Auswahlprozess verwendet werden, um Artefakte miteinander zu vergleichen. Für dieses Beispiel wird allerdings auf einen Vergleich verzichtet, und stattdessen die Ergebnisse analysiert. Die verwendeten Gewichtungen sollten lediglich das Bewertungsverfahren demonstrieren, so dass für diese Analyse die ungewichteten Ergebnisse verwendet werden. Die Ergebnisse lassen sich übersichtlich in einem Netzdiagramm darstellen (siehe Grafik 1).

Es lässt sich erkennen, dass die Hardware im Vergleich zu den übrigen (Software-) Ebenen schlecht abschneidet. Dies mag erstmal verwundern, wurde doch das Ent-

Schicht	Sponsoring	Kompatibilität	Interoperabilität	Verfügbarkeit	Patente	Gesamt
Dateiformat	9	3	1,5	7,8	6,3	27,6
Anwendung	6	6	3	5,2	4,2	24,4
Betriebssystem	9	9	4,5	7,8	4,2	34,5
BIOS	3	1	0,5	2,6	2,1	5,6
Hardware	1,5	1	1	0	2,1	5,6
Gesamt	28,5	20	10,5	23,4	18,9	101,3

Tabelle 9: Gewichtete Ergebnisse aufgrund exemplarischer Gewichtungen

wicklungsmodell des *XO* weithin als gemeinschaftlich, nicht gewinnorientiert und auf Open Source basierend wahrgenommen. Hier wirkt sich insbesondere aus, dass der *XO* wenig bestehende Standards verwendet. Des Weiteren fällt auf, dass das verwendete Dateiformat *ABW* nicht optimal ist, obwohl es seinen Ursprung in einer unkommerziellen Open-Source-Community hat. Obgleich die Spezifikation als Open Source verfügbar ist, handelt es sich um ein proprietäres Format, welches nicht standardisiert und daher inkompatibel zu anderen Applikationen ist. Ebenso gibt es beim *LinuxBIOS* Verbesserungspotential. Besser schneidet *AbiWord* und *GNU/Linux* ab, da es sich um teilweise weit verbreitete Open-Source-Software handelt.

4 Schluss

Trotz des Fehlens einer Definition von Offenheit ist es gelungen, fünf Kriterien zu identifizieren, die Offenheit im Sinne von Benutzerunabhängigkeit charakterisieren und generisch auf unterschiedliche Artefakte anwendbar sind. Diese Kriterien bilden die Grundlage für den *IT Openness Benchmark*; ein abstraktes Bewertungsmodell, welches mittels Stift und Papier angewendet werden kann. Um eine einfache Evaluation und Darstellung zu ermöglichen, steht zusätzlich eine web-basierte Implementierung zur Verfügung.[15] Die Webanwendung enthält bereits bewertete Charakteristiken, z. B. mehrere Open-Source-Lizenzen, zur Auswahl bereit, und unterstützt dadurch eine einfache Evaluierung. Das Ergebnis wird automatisch berechnet und sowohl numerisch als auch grafisch dargestellt.

Es lässt sich zusammenfassen, dass ITOB ein einfach zu benutzendes Verfahren ist, um den Mehrwert der Offenheit an unterschiedlichen Arten von IT-Artefakten zu messen. Dies erlaubt es auch Personen ohne entsprechendes Fachwissen, Computerartefakte miteinander zu vergleichen und offene Artefakte zu identifizieren. Ist der Mehrwert offener Artefakte erkennbar, kann dies Entscheidungen zugunsten offener Artefakte beeinflussen. Es bleibt offen, wie dieser Effekt in der Praxis ausfallen wird.

15 Siehe http://www.opennessbenchmark.info [23. Nov. 2007].

Literatur

Brock, G. W. und Roston, G. L. (Hrsg.) (1996), *The Internet and Telecommunications Policy: Selected Papers From the 1995 Telecommunications Policy Research Conference*, Lawrence Erlbaum, Philadelphia, PA.

Bues, M. (1995), *Offene Systeme: Strategien, Konzepte und Techniken für das Informationsmanagement*, Springer, Berlin.

Cooper, M. N. (2004), Making the Network Connection, *in* M. N. Cooper (Hrsg.), 'Open Architecture as Communications Policy – Preserving Internet Freedom in the Broadband Era', Center For Internet And Society Stanford Law School. http://cyberlaw.stanford.edu/blogs/cooper/archives/openarchitecture.pdf [24. Jan. 2008].

Ensmenger, N. L. (2004), 'Open Source's Lessons for Historians', *IEEE Annals of the History of Computing* **26**(4), S. 103–104.

Free Software Foundation (2007), 'What is Copyleft?', http://www.gnu.org/copyleft/copyleft.html [24. Jan. 2008].

Freyermuth, G. S. (2007), Offene Geheimnisse – Die Ausbildung der Open-Source-Praxis im 20. Jahrhundert, *in* B. Lutterbeck, R. A. Gehring und M. Bärwolff (Hrsg.), 'Open Source Jahrbuch 2007 - Zwischen freier Software und Gesellschaftsmodell', Lehmanns, Berlin, S. 17–57.

Gehring, R. A. und Lutterbeck, B. (2003), 'Software-Patente im Spiegel von Softwareentwicklung und Open Source Software', http://ig.cs.tu-berlin.de/ma/rg/ap/2003-x/GehringLutterbeck-SWPat-092003.pdf [24. Jan. 2008].

Gooch, R. (2003), Requirements on DRM systems, *in* 'Digital Rights Management. Technological, Economic, Legal and Political Aspects', Springer, Berlin, S. 16–25.

Gray, P. A. (1993), *Open Systems – Offene Systeme als Unternehmensstrategie*, McGraw-Hill, London.

Lessig, L. (1999), *Code and other Laws of Cyberspace*, Basic Books, New York.

Merten, S. und Meretz, S. (2005), Freie Software und Freie Gesellschaft, *in* M. Bärwolff, R. A. Gehring und B. Lutterbeck (Hrsg.), 'Open Source Jahrbuch 2005 - Zwischen Softwareentwicklung und Gesellschaftsmodell', Lehmanns, Berlin, S. 293–310.

Open ePolicy Group (2005), Roadmap for Open ICT Ecosystems, techn. Bericht, Berkman Center for Internet and Society at Harvard Law School. http://cyber.law.harvard.edu/epolicy/roadmap.pdf [24. Jan. 2008].

Suhr, J. (2007), Openness Measurement of IT Artifacts, Master's thesis, Technische Universität Berlin, Fakultät IV – Elektrotechnik und Informatik, Institut für Wirtschaftsinformatik und Quantitative Methoden, Fachgebiet Informatik und Gesellschaft. http: //user.cs.tu-berlin.de/~jansuhr/itob/JanSuhr-OpennessMeasurementofITArtifacts.pdf [24. Jan. 2008].

Tapscott, D. und Caston, A. (1993), *Paradigm Shift: The New Promis of Information Technology*, McGraw-Hill, New York.

Varian, H. und Shapiro, C. (2007), Die Ökonomie der Softwaremärkte, *in* B. Lutterbeck, R. A. Gehring und M. Bärwolff (Hrsg.), 'Open Source Jahrbuch 2007 - Zwischen freier Software und Gesellschaftsmodell', Lehmanns, Berlin, S. 125–130.

W3C (2004), 'W3C Patent Policy', http://www.w3.org/Consortium/Patent-Policy-20040205/#sec-Requirements [24. Jan. 2008].

Wheeler, T. (1993), *Offene Systeme. Ein grundlegendes Handbuch für das praktische DV-Management*, Vieweg, Braunschweig/Wiesbaden.

Die deutsche Rechtsprechung zur GNU General Public License

HENRIETTE PICOT

Dieser Artikel befasst sich mit der bisherigen deutschen Rechtsprechung zur Wirksamkeit der Version 2 der *GNU General Public License* (GPL). Seit der deutschland- wie weltweit ersten Gerichtsentscheidung zur Durchsetzbarkeit der *GPL v2* aus dem Jahr 2004 sind eine Reihe weiterer instanzgerichtlicher Entscheidungen ergangen, deren Ergebnisse und Auswirkungen auf die Praxis dieser Artikel zusammenfasst. Über die erst jüngst eingeführte Version 3 der *GNU General Public License* wurde gerichtlich noch nicht entschieden. Angesichts des strukturellen Gleichlaufs zwischen der *GPL v2* und der *GPL v3* ist aber zu erwarten, dass sich die bisherigen Rechtsprechungsergebnisse weitgehend auch auf die *GPL v3* übertragen lassen.

Schlüsselwörter: GPL · Haftungsausschluss · AGB-Recht

1 Einleitung

Im Jahr 2004 hat das Landgericht München I in einer viel beachteten Grundsatzentscheidung[1] als weltweit erstes Gericht über die Wirksamkeit der Version 2 der *GNU General Public License (GPL v2)* entschieden. Das Landgericht München I bestätigte in dieser Entscheidung die grundsätzliche Wirksamkeit der *GPL v2* als Lizenzvertrag zwischen Software-Entwicklern und Nutzern (bzw. Weiterentwicklern). Seitdem haben sich in unterschiedlichen Fallkonstellationen auch das Landgericht Frankfurt[2]

1 LG München I, Az. 21 O 6123/04 – Computer und Recht 2004, S. 774 ff. (Welte./.Sitecom Deutschland GmbH).

2 LG Frankfurt a. M., Az. 2-6 O 224/06 – Computer und Recht 2006, S. 729 ff. (Welte./.D-Link Deutschland GmbH).

sowie das Landgericht Berlin[3] und – erneut – das Landgericht München I[4] mit der Wirksamkeit der *GPL v2* befasst.

Vor diesem Hintergrund soll dieser Artikel dem Leser einen Überblick über die wesentlichen Grundzüge und Ergebnisse der bisherigen deutschen Rechtsprechung zur *GPL v2* geben.

2 Durchsetzbarkeit der GPL v2 und ihrer Kernbestimmungen

2.1 Vertragsschluss

Neben dem Landgericht München I gingen auch das Landgericht Frankfurt und das Landgericht Berlin davon aus, dass zwischen den Urhebern, d. h. den Software–Entwicklern, und den Nutzern von GPL v2-lizenzierter Software ein Lizenzvertrag zustande kommt, in den die Bestimmungen der *GPL v2* als Allgemeine Geschäftsbedingungen wirksam einbezogen werden. Wie jeder Vertragsschluss erfordert dabei auch eine Vereinbarung über die Geltung der *GPL v2* eine vertragliche Einigung (Angebot und Annahme) zwischen den beteiligten Parteien. Die Parteien des Lizenzvertrags sind der/die Software-Entwickler einerseits und jeder einzelne Nutzer (und zugleich ggf. Weiterentwickler) andererseits. Da eine persönliche Kontaktaufnahme zwischen diesen Parteien jedoch in den seltensten Fällen stattfindet, gehen die Gerichte von folgender Konstruktion aus:

Indem ein Software-Entwickler eine bestimmte Software unter die Geltung der *GPL v2* stellt, gibt er ein „an jedermann" gerichtetes Angebot ab, ein Nutzungsrecht an der Software unter den Bedingungen der *GPL v2* zu erwerben. Dabei verzichtet der Software-Entwickler darauf, über die Annahme seines Angebots (durch den/die einzelnen Nutzer) informiert zu werden. Diese besondere Art des Vertragsschlusses ist in § 151 des deutschen Bürgerlichen Gesetzbuchs (BGB) gesetzlich verankert.

Jeder Nutzer, der ein Nutzungsrecht an der der *GPL v2* unterstellten Software erwerben möchte, muss dann ausdrücklich oder „konkludent" (d. h. durch die Nutzung der Software) sein Einverständnis mit der Geltung der *GPL v2* erklären und dadurch das Angebot des Software-Entwicklers annehmen. Indem der Software-Entwickler darauf verzichtet hat, von der Annahme seines Angebots (auf Abschluss eines Lizenzvertrages) unterrichtet zu werden, kommt bereits mit der tatsächlichen Nutzung von GPL v2-lizenzierter Software ein wirksamer Lizenzvertrag zwischen dem Nutzer und dem Software-Entwickler zustande. Mit anderen Worten: Die Bedingungen der *GPL v2* sind also wirksam vereinbart, sobald ein Nutzer unter ihrer ausdrücklichen oder konkludenten Anerkennung beginnt, die GPL v2-lizenzierte Software zu nutzen.

Die Rechtsprechung geht davon aus, dass eine Einigung über die Wirksamkeit der *GPL v2* wie geschildert zustande kommt und der Nutzer daher zur Einhaltung der

3 LG Berlin, Az. 16 O 134/06 – Computer und Recht 2006, S. 735 (WLAN-Router).

4 LG München I Az. 7 O 5245/07 – Computer und Recht 2008, S. 55 ff. (Welte./.Skype) – nicht rechtskräftig.

Bedingungen der *GPL v2* verpflichtet ist. Das von der Rechtsprechung unterstellte Einverständnis des jeweiligen Nutzers mit den Bedingungen der *GPL v2* wird in der juristischen Literatur allerdings als reine Fiktion kritisiert, wenn es für ein solches Einverständnis keine konkreten Anhaltspunkte gibt.[5] In der Tat sind Konstellationen vorstellbar, in denen dem Nutzer die Anwendbarkeit der *GPL v2* auf die von ihm verwendete oder vertriebene Software nicht bekannt ist, so dass man von einem echten Einverständnis mit der Geltung der *GPL v2* kaum ausgehen kann.

Wenn der Nutzer die Bedingungen der *GPL v2* weder ausdrücklich noch konkludent anerkennt, erwirbt er kein Nutzungsrecht an der Software und darf sie daher auch nicht nutzen. Der Inhaber des Urheberrechts an der GPL v2-lizenzierten Software kann dann zwar nicht wegen einer Verletzung der *GPL v2* gegen den Nutzer vorgehen. Er kann aber – ebenso wie im Bereich proprietärer Software – eine Verletzung seines Urheberrechts selbst geltend machen. Mit anderen Worten: Ein Entwickler, der Software unter den Bedingungen der *GPL v2* öffentlich zugänglich macht, verzichtet weder auf sein Urheberrecht als solches noch auf einzelne urheberrechtlich geschützte Positionen. Das hat auch die Rechtsprechung wiederholt bestätigt.[6]

2.2 GPL v2 als Allgemeine Geschäftsbedingungen

Da die *GPL v2* für eine Vielzahl von Lizenzverträgen vorformuliert ist und ihr Inhalt zwischen Entwicklern und Nutzern nicht individuell verhandelt wird, sind die Regelungen der *GPL v2* nach deutschem Recht als *Allgemeine Geschäftsbedingungen* (AGB) einzuordnen (§§ 305 ff. BGB).

Einbeziehung

Allgemeine Geschäftsbedingungen sind nur dann wirksam vereinbart, wenn der Nutzer auf ihre Geltung hingewiesen wird und es ihm möglich ist, „auf zumutbare Weise Kenntnis von ihrem Inhalt zu nehmen". Da der Volltext der *GPL v2* im Internet leicht aufzufinden und abrufbar ist, hat die Rechtsprechung an einer wirksamen Einbeziehung keinen Zweifel, sofern der Nutzer auf ihre Geltung eindeutig hingewiesen wird.[7]

Inhaltliche Anforderungen

Als *Allgemeine Geschäftsbedingungen* unterliegen die Regelungen der *GPL v2* strengeren gesetzlichen Anforderungen als Regelungen in Lizenzverträgen, die zwischen

5 Vgl. Grützmacher, Anmerkung zu LG Frankfurt a. M., Az. 2-6 O 224/06 – Computer und Recht 2006, S. 733 ff.

6 Vgl. LG München I – Computer und Recht 2004, 774 (775); LG Frankfurt a. M. – Computer und Recht 2006, 729 (730); LG Berlin – Computer und Recht 2004, S. 735.

7 Vgl. LG München I – Computer und Recht 2004, S. 774 f.; LG Frankfurt a. M. – Computer und Recht 2006, S. 729 f.

ihren Parteien individuell ausgehandelt werden. *Allgemeine Geschäftsbedingungen* dürfen den sog. Verwendungsgegner (d. h. die Vertragspartei, der die Allgemeinen Geschäftsbedingungen von ihrem Vertragspartner vorgegeben werden – im Falle der GPL ist dies der Software-Nutzer) nicht „unangemessen benachteiligen". Sie müssen ferner klar und eindeutig formuliert und gestaltet sein (sog. Transparenzgebot) und dürfen keine Klauseln enthalten, die nach den jeweiligen Umständen so ungewöhnlich sind, dass der Verwendungsgegner mit ihnen nicht zu rechnen braucht.

Mit Blick auf das deutsche AGB-Recht erscheint insbesondere die Durchsetzbarkeit von § 4 Satz 2 der *GPL v2* fraglich: Danach soll das gemäß der *GPL v2* eingeräumte Nutzungsrecht automatisch enden, wenn ein Nutzer gegen die Bedingungen der *GPL v2* verstößt. Bemerkenswerterweise hat die bisherige (instanzgerichtliche) Rechtsprechung diesen Mechanismus für wirksam gehalten. Sie hält dieses automatische Erlöschen von Nutzungsrechten auch nicht für eine unangemessene Benachteiligung des Nutzers.[8]

Die *GPL v2* enthält in §§ 11, 12 einen umfassenden Haftungsausschluss hinsichtlich der Qualität und Leistungsfähigkeit der Software. Jeder Nutzer, der GPL v2-lizenzierte Software an Dritte weitergibt, muss diesen Haftungsausschluss gemäß § 1 der *GPL v2* außerdem ausdrücklich an die dritten Softwareempfänger weitergeben. Deutsche Gerichte haben zwar über diesen Haftungsausschluss bisher noch nicht geurteilt. Nach deutschem Recht können in den Allgemeinen Geschäftsbedingungen aber weder die Haftung für Vorsatz noch die Haftung für grobe Fahrlässigkeit durch den Verwender der Allgemeinen Geschäftsbedingungen, hier also den Software-Entwickler, ausgeschlossen werden, und auch die Haftung für einfache oder leichte Fahrlässigkeit ist nur in sehr eingeschränktem Umfang begrenzbar. Diesen Anforderungen werden die §§ 11, 12 und 1 der *GPL v2* nicht gerecht. Es ist daher mit an Sicherheit grenzender Wahrscheinlichkeit davon auszugehen, dass der in diesen Regelungen der *GPL v2* vorgesehene Haftungsausschluss einer gerichtlichen Überprüfung in Deutschland nicht standhalten würde.

2.3 Pflichten des Nutzers gemäß §§ 1, 2 und 3 der GPL v2

Mit der Annahme der wirksamen Einbeziehung der *GPL v2* sowie mit der Anerkennung einiger ihrer Kernbedingungen hat die Rechtsprechung das Konzept der *GPL v2* im Grundsatz bestätigt. Dies gilt insbesondere für die Verpflichtung des Nutzers, GPL v2-lizenzierte Programme nur unter Offenlegung des Quellcodes und unter Geltung der Bedingungen der *GPL v2* zu verbreiten (§ 3 der *GPL v2*). Die Rechtsprechung hat ferner bestätigt, dass mit jeder an einen Dritten weitergegebenen Kopie von gemäß GPL v2-lizenzierter Software auch eine Kopie des Textes der *GPL v2* weitergegeben werden muss. Dagegen hat die Rechtsprechung über die automatische „GPL v2-Infizierung" von Programmen, die auf GPL v2-lizenzierter

8 Näheres unter Ziffer 2.4.

Software basieren und durch eine Bearbeitung derselben erstellt wurden (sog. „viraler Effekt") bisher nicht entschieden.

Verschaffung einer Kopie der GPL v2

Gemäß § 1 der *GPL v2* ist der Nutzer (unter anderem) dazu verpflichtet, jedem Dritten, dem er eine Kopie der Software verschafft, auch eine Kopie der *GPL v2* zukommen zu lassen. Das gilt unabhängig davon, ob die von dem Nutzer weitergegebene Kopie die Software in ihrer Ausgangsfassung oder in einer vom Nutzer in Übereinstimmung mit § 2 bearbeiteten Fassung enthält.

Die deutsche Rechtsprechung erkennt die Wirksamkeit dieser Pflicht zur Verschaffung einer Kopie der *GPL v2* jedenfalls für den Fall einer Weitergabe der Ausgangsfassung grundsätzlich an und nimmt sie sehr genau. In einer erst jüngst – im Juli 2007 – ergangenen Entscheidung hat das Landgericht München I diese Anforderung bestätigt.[9] In diesem Urteil gegen das Luxemburger Unternehmen *Skype* befand das Landgericht München I den folgenden Hinweis auf die Anwendbarkeit der *GPL v2* und der *Lesser GNU General Public License* (LGPL) für nicht ausreichend:

> „This product includes software code developed by third parties, including software code subject to the GNU General Public License („GPL") or GNU Lesser General Public License („LGPL"). As applicable, the terms of the GPL and LGPL, and information on obtaining access to the GPL Code and LGPL Code used in this product, are available to you at http://www.smc.com under the support/downloads section. . . . For details, see the GPL code and LGPL Code for this product and the terms of the GPL and LGPL."

Nach Ansicht des Landgerichts München I genügt dieser Hinweis den Anforderungen gemäß § 1 der *GPL v2* aus zwei Gründen nicht: Zum einen ermöglicht er dem Erwerber der Software nur den Download von der angegebenen Internetseite, wohingegen § 1 der *GPL v2* die Weitergabe einer „Kopie" der Lizenzbedingungen selbst erfordert. Zum anderen ist durch den Bezug sowohl auf die GPL als auch auf die LGPL unklar, für welche Teile des vertriebenen Produkts – im konkreten Fall ging es um eine VoIP-Telefonsoftware – die GPL und für welche Produktteile die LGPL gelten sollte. Das hätte nach Auffassung des Landgerichts München genauer dargestellt werden müssen.

Zugänglichmachung des Quellcodes

Gemäß § 3 der *GPL v2* ist der Nutzer verpflichtet, jedem Dritten, dem er eine Kopie von GPL v2-lizenzierter Software (oder eine von ihm gemäß § 2 erstellte Bearbeitung)

9 LG München I Az. 7 O 5245/07 – Computer und Recht 2008, 55 ff. (Welte./.Skype) – nicht rechtskräftig.

im Objektcode weitergibt, auch den Quellcode der weitergegebenen Software zugänglich zu machen. Der Nutzer kann diese Verpflichtung auf verschiedene Weise erfüllen (z. B. durch ein mindestens drei Jahre gültiges schriftliches Angebot zur Verschaffung einer vollständigen maschinentextlesbaren Kopie des Quelltextes)[10]. Für den Fall der Weitergabe einer Kopie der GPL v2-lizenzierten Software hat die Rechtsprechung die Durchsetzbarkeit dieser Verpflichtung bestätigt.

Dabei hat das Landgericht München I in seiner bereits erwähnten Entscheidung aus Juli 2007[11] zwischen den verschiedenen Möglichkeiten der Zurverfügungstellung des Quellcodes sehr genau differenziert. Unter Verweis auf § 3 Absatz 3 der *GPL v2* hielt das Gericht den bloßen Hinweis auf die URL einer Internetseite, auf der der Quellcode zum Download bereitgehalten wird, nur dann für ausreichend, wenn auch der Objektcode nur auf diese Weise erhältlich ist. Da der Objektcode in dem vom Landgericht München I zu entscheidenden Fall aber in ein VoIP-Telefon integriert und nicht per Download aus dem Internet erhältlich war, stellte das Gericht eine Verletzung von § 3 der *GPL v2* fest.[12]

2.4 Automatische Beendigung des Nutzungsrechts bei Verstoß gegen GPL v2

Gemäß § 4 Satz 2 der *GPL v2* endet das Nutzungsrecht automatisch, wenn der Nutzer die GPL v2-lizenzierte Software entgegen den Bedingungen der *GPL v2* vervielfältigt, weiterlizenziert, modifiziert oder verbreitet. Wie oben erwähnt, hat die Rechtsprechung die Wirksamkeit dieses Mechanismus bestätigt und betrachtet ihn nicht als eine unangemessene Benachteiligung des Nutzers.[13]

Die Rechtsprechung stellte zunächst fest, dass § 4 Satz 2 der *GPL v2* keine dinglich wirkende Nutzungsrechtsbeschränkung darstellt (d. h. eine Beschränkung des Umfangs des eigentlichen Nutzungsrechts im Sinne des § 31 UrhG), sondern lediglich vertragliche Wirkung entfaltet.[14] Dinglich wirkende Beschränkungen eines Nutzungsrechts sind nach dem deutschen Urheberrecht nur zum Zwecke der Aufspaltung des Nutzungsrechts in abgrenzbare, wirtschaftlich-technisch einheitliche und selbstständige Nutzungsarten möglich.[15] Diese Voraussetzungen erfüllen die in §§ 2, 3 der *GPL v2* festgelegten Bedingungen der Vervielfältigung, Weiterlizenzierung, Modifi-

10 Vgl. § 3 Absatz Nr. 1-3 sowie Absatz 3 der GPL v2.

11 Vgl. oben Ziffer 2.3.1. sowie LG München I – Computer und Recht 2008, S. 55 ff.

12 LG München I – Computer und Recht 2008, S. S. 55 (59).

13 Vgl. LG München I – Computer und Recht 2004, S. 775 (776); LG Frankfurt – Computer und Recht 2006, S. 729 (731).

14 Der Unterschied zwischen einer dinglichen und einer nur vertraglichen Beschränkung eines Nutzungsrechts wird deutlich, wenn sich der Nutzer an die jeweilige Beschränkung nicht hält: Im Falle einer dinglich wirkenden Beschränkung des eingeräumten Nutzungsrechts könnte der Entwickler gegen den Nutzer nicht nur wegen einer Verletzung der *GPL v2* vorgehen, sondern er wäre unmittelbar in seinem eigenen Urheberrecht verletzt. Eine rein vertraglich wirkende Beschränkung des Nutzungsrechts berechtigt den Entwickler dagegen nur zur Geltendmachung vertraglicher Ansprüche.

15 Diese Grundsätze hat der BGH in seiner sog. „OEM-Entscheidung" festgelegt, vgl. BGH GRUR 2001, S. 153 (154).

zierung oder Verbreitung, deren Nichtbeachtung den Rechterückfall gemäß § 4 Satz 2 der *GPL v2* auslösen soll, nicht.

Allerdings ist § 4 Satz 2 der *GPL v2* nach Auffassung der Rechtsprechung als eine auflösende Bedingung der Nutzungsrechtseinräumung zu verstehen (§ 158 BGB). Damit ist sie nach Auffassung der Rechtsprechung wirksam. Der Entwickler räumt dem Nutzer das Nutzungsrecht an der GPL v2-lizenzierten Software also nur unter der Bedingung ein, dass der Nutzer die Bedingungen der *GPL v2* einhält. Verstößt der Nutzer gegen die *GPL v2*, indem er die Software entgegen der *GPL v2* vervielfältigt, weiterlizenziert, modifiziert oder verbreitet, so tritt die Bedingung ein und führt zur automatischen Beendigung des Nutzungsrechts. Dieser Rechterückfall lässt Nutzungsrechte Dritter, die von dem GPL v2-brüchigen Nutzer zuvor „Kopien oder Rechte" unter der *GPL v2* erhalten haben, gemäß § 4 Satz 3 der GPL aber unberührt, solange diese Dritten selbst die Bedingungen der *GPL v2* anerkennen und befolgen. Auch dem Nutzer selbst ist es nach einem Verstoß gegen die *GPL v2* unbenommen, unter Anerkennung und Befolgung der *GPL v2* jederzeit ein neues Nutzungsrecht zu erwerben. Da die Wirkung des § 4 Satz 2 somit andere, „GPL v2-treue" Teilnehmer des Rechtsverkehrs nicht beeinträchtigt und auch den Nutzer selbst nicht auf Dauer benachteiligt (sofern er die Bedingungen der *GPL v2* künftig befolgt), hat die Rechtsprechung den automatischen Rechtsverlust gemäß § 4 Satz 2 im Ergebnis für wirksam gehalten.

Für die Praxis bedeutet dies, dass jeder Nutzer, der GPL v2-lizenzierte Software entgegen den Bedingungen der *GPL v2* vervielfältigt, weiterlizenziert, modifiziert oder verbreitet, eine Urheberrechtsverletzung begeht. Etwaige dritte Empfänger (etwa Kunden) können solche Software aber dennoch berechtigt nutzen, sofern sie sich selbst an die Bedingungen der *GPL v2* halten.

3 Wer kann eine Verletzung der GPL v2 gerichtlich geltend machen?

3.1 Urheberschaft an einem Softwareprogramm

Da Open-Source-Software typischerweise das Ergebnis gemeinsamer Entwicklungsarbeit einer Vielzahl von Software-Entwicklern ist, stellt sich die Frage, wer eigentlich dazu berechtigt ist, gerichtliche Schritte gegen Verletzungen der *GPL v2* einzuleiten. Unabhängig davon, von welcher Mittelsperson ein Nutzer die jeweilige Open–Source-Software tatsächlich erhält, wird ihm sein Nutzungsrecht stets unmittelbar von dem/den Entwickler(n) der Software eingeräumt. Die Bedingungen der *GPL v2* gelten dabei immer unmittelbar zwischen dem/den Entwickler(n) und dem Nutzer (und nicht etwa zwischen dem die Software verbreitenden Mittelsmann und dem Nutzer). Folglich steht auch das Recht zur Verfolgung von Verletzungen der *GPL v2* grundsätzlich dem/den Entwickler(n) zu.

Nach der Systematik des deutschen Urheberrechts gelten mehrere Entwickler eines Softwareprogramms entweder jeweils als Alleinurheber eines bestimmten Programmteils (§ 7 UrhG) oder als Miturheber des Gesamtprogramms (§ 8 UrhG). Diese Einordnung hängt davon ab, ob und in welchem Maße die Entwickler im Einzelfall gemeinschaftlich an der Verwirklichung einer Gesamtidee mitgewirkt und ihre eigene Entwicklung einer Gesamtidee untergeordnet haben. Haben die Entwickler zeitlich gestaffelt jeweils einen eigenen Entwicklungsschritt erzielt, der zwar auf dem vorangegangenen Schritt aufbaut, jedoch einen eigenständigen und in seinen Funktionalitäten vorab nicht gemeinschaftlich konzipierten Programmteil darstellt, so dürfte eine Alleinurheberschaft an den jeweiligen einzelnen Programmteilen naheliegen. Waren die einzelnen Entwicklungsschritte hingegen in ein gemeinsames Gesamtkonzept planmäßig eingebettet, wie es im Rahmen von Open-Source-Projekten häufig der Fall ist, so werden die Entwickler eher als Miturheber gemäß § 8 UrhG anzusehen sein. In den bisherigen Entscheidungen zur *GPL v2* hat die Rechtsprechung die Frage der Allein- oder Miturheberschaft der Entwickler nicht selbst beurteilt, da sie entweder aufgrund der gesetzlichen Urhebervermutung (siehe dazu Ziffer 3.2) oder des (vom Prozessgegner nicht bestrittenen) Klägervortrags entscheiden konnte.[16]

Die Qualifizierung als Allein- oder Miturheber spielt bei der Rechtsdurchsetzung eine wichtige Rolle, da ein Alleinurheber die Verletzung des von ihm entwickelten Programmteils unabhängig von den anderen Urhebern verfolgen kann. Ein Miturheber dagegen kann eine Rechtsverletzung des Gesamtprogramms zwar gerichtlich geltend machen, muss aber (als sog. Prozessstandschafter) das gemeinschaftliche Urheberrecht aller Miturheber geltend machen und kann daher auch vom Beklagten nur Leistung an alle Miturheber verlangen (§ 8 Absatz 2 UrhG).

3.2 Gesetzliche Urhebervermutung

Eine wichtige Rolle spielt in diesem Zusammenhang die gesetzliche Urhebervermutung gemäß § 10 UrhG. Danach wird vermutet, dass eine Person, die im Quellcode eines Softwareprogramms als Urheber bezeichnet ist, auch tatsächlich der Urheber des betreffenden Programms ist. Sind mehrere Personen in dieser Weise bezeichnet, so wird vermutet, dass diese Personen als Miturheber tätig waren. Soweit aufgrund eines Urhebervermerks im Quellcode die Urheberrechtsvermutung greift, müsste der wegen einer Urheberrechtsverletzung Beklagte beweisen, dass die im Quellcode als Urheber bezeichnete(n) Person(en) nicht der wirkliche Urheber des Programms ist bzw. sind.[17]

16 Vgl. insbesondere LG Frankfurt – Computer und Recht 2006, S. 729 (730).

17 Vgl. etwa LG Frankfurt a. M., Az. 2-6 O 224/06 – Computer und Recht 2006, S. 729 (730), (Welte./.D-Link Deutschland GmbH).

3.3 Treuhänderische Einräumung ausschließlicher Nutzungsrechte

Sowohl das Landgericht Frankfurt als auch das Landgericht München I[18] haben in ihren Entscheidungen anerkannt, dass eine Verletzung der Bedingungen der GPL nicht nur von dem/den Entwickler(n) selbst, sondern auch von einem ausschließlichen (d. h. exklusiven) Nutzungsrechtsinhaber geltend gemacht werden kann.

Mit dieser Konstruktion einer treuhänderischen, ausschließlichen Nutzungsrechtseinräumung hatte Harald Welte, der Gründer der Organisation *gpl-violations.org*[19], in mehreren Fällen Erfolg. Die Organisation *gpl-violations.org* hat sich die Durchsetzung der Bedingungen der GPL zum Ziel gesetzt. Sie hat in den letzten Jahren in zahlreichen Fällen außergerichtlich (durch Abmahnungen) und in einigen Fällen gerade in Deutschland auch gerichtlich (durch Unterlassungs- und/oder Schadensersatzklagen) Verletzungen der Bedingungen der GPL erfolgreich verfolgt.

Harald Welte machte in diesen Fällen Verletzungen bestimmter Bestandteile des Linux-Kernels geltend, deren Entwickler Harald Welte die ausschließlichen (d. h. exklusiven) Rechte der Vervielfältigung, Verbreitung und öffentlichen Wiedergabe sowie das Recht eingeräumt hatten, „Dritten die Vorname von Bearbeitungen und Ergänzungen zu gestatten". Die Gewährung ausschließlicher Nutzungsrechte an einem Softwareprogramm ist rechtlich auch dann noch möglich, wenn der/die Entwickler – zum Beispiel unter der *GPL v2* – zuvor Dritten nicht-ausschließliche (einfache) Nutzungsrechte eingeräumt hat/haben.[20]

4 Wer kann wegen einer Verletzung der GPL v2 verklagt werden?

In der Mehrzahl der von der Rechtsprechung bisher entschiedenen Fälle entsprach die Sachlage dem „klassischen" Lizenzverletzungsszenario: Der Beklagte/Antragsgegner hatte beim Vertrieb GPL v2-lizenzierter Software Lizenzbedingungen der *GPL v2* nicht beachtet (z. B. bei der Weitergabe von GPL v2-lizenzierter Software den Quellcode nicht nach Maßgabe der *GPL v2* zur Verfügung gestellt oder keine Kopie der *GPL v2* mitgeliefert) und wurde als Verletzer auf Unterlassung und/oder Schadensersatz in Anspruch genommen.[21]

Dagegen war in der bereits erwähnten jüngsten Entscheidung des Landgerichts München I der Antragsgegner nicht der Anbieter des unter Verstoß gegen die *GPL v2* vertriebenen VoIP-Telefons (*SMC Networks*), sondern der Betreiber der Plattform,

18 Vgl. Fußnoten 1, 2, 4. Bei den Entscheidungen des Landgerichts München I handelte es sich allerdings nicht um Endurteile, sondern um einstweilige Verfügungen, für deren Erlass die Berechtigung des Klägers zur gerichtlichen Verfolgung der in Frage stehenden Rechtsverletzungen (so genannte Aktivlegitimation) nicht bewiesen, sondern lediglich glaubhaft gemacht werden muss.

19 Siehe http://www.gpl-violations.org [10. Feb. 2008].

20 Vgl. § 33 Satz 1 UrhG. Der Inhaber des einfachen Nutzungsrechts genießt insoweit sog. „Sukzessionsschutz".

21 So in LG München I – Computer und Recht 2004, S. 774 ff.; LG Frankfurt – Computer und Recht 2006, 729 ff.; LG Berlin – Computer und Recht 2004, S. 735 ff.

auf der das VoIP-Telefon angeboten wurde (*Skype*). Das Landgericht München I meinte, dass *Skype* als Plattformbetreiber den nicht GPL v2-konformen Vertrieb spätestens ab dem Zeitpunkt hätte unterbinden müssen, ab dem *Skype* von den die *GPL v2* verletzenden Umständen wusste. Das Landgericht München I erließ daher eine einstweilige Verfügung gegen *Skype*.

Harald Weltes gerichtlichem Antrag auf einstweilige Verfügung gegen *Skype* war eine außergerichtliche Abmahnung vorausgegangen. Als Reaktion auf die Abmahnung hatte *SMC Networks* dem – bis dahin ohne jeglichen Hinweis auf die GPL vertriebenen – VoIP-Telefon ein Beiblatt mit einem allgemeinen Hinweis auf die GPL und die LGPL (siehe oben Ziffer 2.3.1) hinzugefügt. Dieser Hinweis genügte aber weder den Anforderungen des § 1 (Kopie der *GPL v2*)[22] noch denen des § 3 der *GPL v2* (Zurverfügungstellung des Quellcodes)[23]. Diese Rechtsverletzung hätte *Skype* nach Ansicht des LG München I unterbinden müssen.

Ob diese Erweiterung der Haftung auf den Plattformbetreiber mit den vom Bundesgerichtshof aufgestellten, eher restriktiven allgemeinen Grundsätzen zur so genannten „Störerhaftung" vereinbar ist, erscheint zweifelhaft und wurde in der juristischen Literatur auch bereits in Frage gestellt.[24] Es bleibt daher abzuwarten, ob die (noch nicht rechtskräftige) Entscheidung des LG München I von anderen Instanzgerichten und/oder höchstrichterlich bestätigt wird.

Anzumerken ist noch, dass auch gegen den Hersteller des VoIP-Telefons, *SMC Networks*, ein gerichtliches Verfahren anhängig ist.

5 Fazit und Ausblick

Sämtliche bisher ergangenen instanzgerichtlichen Entscheidungen bestätigen – zumindest in ihren praktischen Ergebnissen – die im Jahr 2004 vom Landgericht München I in der weltweit ersten Gerichtsentscheidung zur GPL vertretene Auffassung. Trotz nicht immer deckungsgleicher juristischer Begründungen der Gerichte ist in der Tendenz klar, dass deutsche Gerichte die Bedingungen der *GPL v2* im Grundsatz für durchsetzbar halten. Es bleibt allerdings abzuwarten, ob der Bundesgerichtshof, der über die Wirksamkeit der *GPL v2* bisher noch nicht entschieden hat, diese Rechtsprechung der Instanzgerichte bestätigen wird.

Hinsichtlich der erst jüngst eingeführten Version 3 der *GNU General Public License (GPL v3)* gibt es bisher keine Rechtsprechung. Angesichts des strukturellen Gleichlaufs zwischen der *GPL v2* und der *GPL v3* ist jedoch zu erwarten, dass die Gerichte bei einer Beurteilung der *GPL v3* von den für die *GPL v2* erzielten praktischen Ergebnissen nicht wesentlich abweichen werden. Die bisherigen Ent-

22 Vgl. oben Ziffer 2.3.1.

23 Vgl. oben Ziffer 2.3.2.

24 Vgl. Wimmers/Klett, Anmerkung zu LG München I, Az. 7 O 5245/07 – Computer und Recht 2008, 59 (60).

scheidungen werden vielmehr auch mit Blick auf die Anwendung und Auslegung der *GPL v3* von maßgeblicher Bedeutung sein.

Open Source Content Management – Eine kritische Betrachtung

TOBIAS HAUSER UND ANDI PIETSCH

Typo3, *Joomla!*, *Drupal*, *DotNetNuke* und *Plone*, so heißen einige Open–Source-Content-Management-Systeme (kurz *OS-CMS*), die sich nicht nur bei Hobbynutzern, sondern auch in Unternehmen einen Namen gemacht haben. Was ist also dran am Boom der Open-Source-Lösungen? Werden sie auf lange Sicht die kommerziellen Alternativen zumindest in bestimmten Marktbereichen verdrängen? Dieser Beitrag möchte die Pauschalaussagen aus beiden Welten auf den Prüfstand stellen. Anhand umfangreicher Praxiserfahrungen wird untersucht, welche Vor- und Nachteile OS-CMS haben und wie sich diese auf das Projektergebnis auswirken. Außerdem wird der Prozess der Systementscheidung analysiert, da sich hier zeigt, dass wichtige Unterschiede oft nicht nur zwischen Open Source und kommerzieller Software liegen, sondern dass auch innerhalb der vielen OS-CMS große Unterschiede vorhanden sind.

Schlüsselwörter: Content Management System · CMS · DotNetNuke · Plone · Joomla! · Drupal · Typo3

1 CMS – eine Definition

Ein Content-Management-System (CMS) ist in unserem Kontext ein System zur Inhaltsverwaltung von Webseiten. *Inhalt* bezeichnet dabei jede Art von Information, Bildern und Dokumenten in verschiedenen Formaten. Enger gefasst kümmert sich die Inhaltsverwaltung nur um die veränderlichen Inhalte.[1] Der Begriff *Management* umfasst die einfache Verwaltung von Inhalten, die redaktionell erzeugt oder ausgewählt werden sollen (vgl. Rawolle 2002, S. 15). Das heißt ein CMS hat im Allgemeinen zum Ziel, dass auch Bearbeiter (synonym Redakteure) ohne Web- und HTML-Kenntnisse

1 Im Normalfall werden auch sehr selten geänderte Inhalte in einem CMS als veränderliche Inhalte angelegt. Allerdings gibt es auch einfache Lösungen, die nur Teile von Inhalten veränderlich machen.

Inhalte schnell und effizient ändern können. Damit sollen grundlegend die Effizienz und Effektivität des Bereitstellungsprozesses von Inhalten erhöht werden. Der Begriff *System* steht für eine in sich geschlossene Anwendung. Sie kann neben dem Primärziel – der Verwaltung von Inhalten – noch viele Sekundärziele beinhalten, z. B. Erweiterung um Funktionalität wie Kontaktformulare, Newsletter, Suche, Nutzerverwaltung und Personalisierung oder allgemeiner gesprochen eine Erhöhung des Automatisierungsgrades, bei der System- und Medienbrüche vermieden werden und die Flexibilität und Wiederverwertung von Inhalten und Funktionen verbessert wird.

1.1 Offlinelösungen

Die hier als Content-Management-System besprochenen Systeme arbeiten webbasiert. Das heißt die Verwaltungsoberfläche (auch *Backend* genannt) für die Inhalte und damit die Inhaltspflege finden über den Browser statt. Diese Art von Systemen wird in der Literatur auch als *Web Content Management* bezeichnet (vgl. Rawolle 2002, S. 40). Damit erfolgt die Abgrenzung zu Offlinelösungen, bei denen die Inhaltsbearbeitung per Desktop-Programm erfolgt. Eine solche Lösung bietet beispielsweise die Firma *Adobe* mit ihrem Webeditor *Dreamweaver* und dem zugehörigen Werkzeug zur Inhaltsbearbeitung, *Contribute*. Eine Zwischenlösung sind Systeme, bei denen die Liveversion mit einer Offlineversion abgeglichen wird. Allerdings zählen diese Lösungen meist auch zu den Web-Content-Management-Systemen, da beide Versionen über eine Weboberfläche verwaltet werden.

1.2 Abgrenzung zum Redaktionssystem

Das klassische *Redaktionssystem* hat lange Zeit den Arbeitsprozess (auch *Workflow*) einer Printredaktion abgebildet. Dieses System – bestehend aus mehreren Redakteuren, Chefredakteur und gegebenenfalls Schlussredaktion in einem Korrekturprozess – wurde in die Onlinewelt abgebildet. Im Prinzip sind Bearbeiter von Inhalten in einem CMS auch Redakteure. In der Praxis wird ein CMS dann Redaktionssystem genannt, wenn es Möglichkeiten enthält, einen Arbeitsprozess abzubilden. Das heißt im einfachsten Fall kann ein Redakteur den Inhalt nur bearbeiten, aber nicht freischalten. Das System benachrichtigt nach der Bearbeitung seitens des Redakteurs den Chefredakteur. Dieser kann dann den Inhalt weiter editieren, freigeben oder zur Korrektur an den Redakteur zurücksenden. Viele OS-CMS besitzen solche Funktionen – allerdings ist die Ausprägung hier sehr unterschiedlich (siehe auch Abschnitt 3).

1.3 Portal

Ein *Portal* enthält nicht nur Inhalte, sondern vereinigt auch verschiedene Dienste wie *RSS feeds*, Inhalte von Fremdanwendern (z. B. per *Portlet* oder *Web Service*) und eine Vielzahl an Funktionen (vgl. Baumgarten et al. 2001, S. 46 ff.). Oft ist ein Portal auch durch bestimmte Themen gekennzeichnet, z. B. Nachrichtenportale wie

Spiegel Online[2] oder IT-Portale wie *Golem.de*. Moderne CMS sind meist zugleich auch Portale und bieten entsprechende Fähigkeiten, um verschiedene Funktionalitäten vom Forum bis zum *Newsfeed* zu integrieren. Insofern wird in der Open-Source-CMS-Welt im Speziellen und in der CMS-Welt im Allgemeinen softwareseitig kaum zwischen Portal und CMS unterschieden. Manche CMS spezialisieren sich allerdings auf die Portalfunktionen, wie z. B. DotNetNuke, oder betonen bestimmte Fähigkeiten, wie z. B. Drupal mit der Community-Integration.

1.4 Enterprise-CMS

Fast alle heute angebotenen kommerziellen CMS nennen sich *Enterprise-CMS*. Hier gibt es keine exakte Begriffsabgrenzung. Enterprise-CMS zeichnen sich vor allem durch ihre Eignung für den Unternehmenseinsatz aus. Welche Kriterien, von Skalierbarkeit über Workflow bis hin zum Rechtemanagement, dafür aber ausschlaggebend sind, lässt sich kaum festlegen. Wie Abschnitt 3 zeigt, besitzt hier auch im OS-Bereich jedes System individuelle Vorzüge.

1.5 Abgrenzung zum Dokumentenmanagement

Das *Dokumentenmanagement* erfolgt zwar auch oft mit einer webbasierten Oberfläche, allerdings arbeitet hier unter der Oberfläche meist ein System, das auf Dateisystembasis Verwaltungsmöglichkeiten anbietet. Ein Dokumenten-Management-System (DMS) hat nicht die Darstellung von Inhalten zum Ziel, sondern soll die Arbeit mit Dokumenten, z. B. in Arbeitsgruppen, vereinfachen. Zwar gibt es in einigen Open-Source-CMS auch entsprechende Erweiterungen (z. B. *DocMan* in Joomla!), diese unterscheiden sich aber deutlich von dateisystembasierten Dokumenten-Management-Lösungen.

2 Vor- und Nachteile von OS-CMS – ein Überblick

Sucht man nach den wichtigsten Eigenschaften von OS-CMS, stößt man sofort auf den Entfall von Lizenzkosten.[3] Außerdem wird der Quelltext mitgeliefert, was eigene Änderungen ermöglicht. In diesem Zusammenhang ist die verwendete Lizenz wichtig – sie schränkt beispielsweise die Entwicklung von kommerziellen Erweiterungen ein. Eine weitere relevante Frage bei OS-CMS ist, wer die Systeme entwickelt. Hier unterscheidet man die Entwickler des Kern-CMS und die Entwickler von Erweiterungen. Die Kernentwickler sind entweder eine Firma (z. B. ursprünglich bei *Mambo* oder bei *Contenido*) oder eine Vereinigung aus relativ wenigen Entwicklern (z. B. bei Typo3). Die Entwickler von Erweiterungen stellen meist eine große Personengruppe

2 http://www.spiegel.de.

3 Auch wenn Open Source nicht gleich Freeware ist und auch nicht immer Lizenzkostenfreiheit bedeutet, sind die meisten OS-CMS lizenzkostenfrei. Siehe auch Abschnitt 2.1.

dar. Mit Hilfe dieser verschiedenen Eigenschaften und den entsprechenden Erkenntnissen aus der Praxis lassen sich die einzelnen Kriterien, die für den Erfolg eines CMS relevant sind, gewichten und so die Vor- und Nachteile von OS-CMS herausarbeiten.

2.1 Lizenz-, Erstellungs- und Implementierungskosten

Lizenz- und Erstellungskosten existieren bei einem OS-CMS im Regelfall nicht, ein OS-CMS ist frei verfügbar und kann von jedem installiert und benutzt werden, ohne dass nachfolgende Kosten für das CMS entstehen. Als Basis für ein Open-Source–System wird von der jeweiligen Entwicklergruppe eine Open-Source-Lizenz[4] gewählt. Die bekannteste Lizenz in diesem Bereich ist die *GNU General Public Licence* (GPL) von der *Free Software Foundation*.[5] Sie erlaubt nicht nur die lizenzfreie Weitergabe, sondern stellt sie über das so genannte *Copyleft* sogar sicher. Das heißt die GPL bestimmt, dass Quelltext, der einen unter der GPL stehenden Quelltext als Basis verwendet, auch unter der GPL stehen muss. Dies gilt beispielsweise für Erweiterungen, die mit der Programmierschnittstelle eines unter der GPL stehenden Open-Source–CMS entwickelt werden und so selbst unter der GPL stehen. Die GPL verbietet dabei nicht die kostenpflichtige Weitergabe, sondern fordert nur, dass der Quelltext mitgeliefert wird und derjenige, der ihn erhält, ihn wiederum ohne Einschränkungen weitergeben kann. Für Unternehmen hat dies keine negativen Auswirkungen, wenn Erweiterungen nur zum Eigenbedarf entwickelt werden. Ein kommerzieller Verkauf von Erweiterungen ist allerdings unmöglich oder bei Projekten wie Joomla! sehr umstritten.[6]

Kommerzielle CMS sind meist mit einer Anfangsinvestition und monatlichen oder jährlichen Lizenzkosten verbunden. Dies sind die Voraussetzungen, um kommerzielle CMS aus rechtlicher Sicht nutzen zu können. Die Arten der Nutzungsbedingungen können sich je nach System allerdings stark unterscheiden. Vor allem ist der Zugriff auf den Quelltext der Applikation oft eingeschränkt. Hier gilt es genau zu prüfen, ob ein CMS-Projekt in Zukunft eigenständige Weiterentwicklungen benötigt und ob die Lizenzbedingungen des jeweiligen Anbieters dies erlauben.

Bei Projekten mit kommerziellen CMS zeigt die Praxis, dass die Entscheidung für ein bestimmtes kommerzielles CMS meist aufgrund von Funktionalitäten getätigt wurde. Die Frage nach den technischen Gegebenheiten im Unternehmen wurde dann mit dem Anbieter des CMS optimiert, was nichts anderes bedeutet, als dass die Voraussetzungen für die Lauffähigkeit des CMS neu angeschafft werden mussten. Bei der Entscheidung für ein OS-CMS steht mehr die Frage der technischen Voraussetzun-

4 Definition einer Open-Source-Lizenz: http://www.opensource.org/docs/definition.php [5. Jan. 2008].
5 Offizieller Lizenztext: http://www.gnu.org/licenses/gpl-3.0.html [5. Jan. 2008], inoffizielle deutsche Übersetzung: http://www.gnu.de/documents/gpl.de.html [5. Jan. 2008].
6 Siehe die Diskussionen unter http://www.joomla.org/component/option,com_jd-wp/Itemid,105/p,371/ [5. Jan. 2008].

gen und der Realisierung im Vordergrund. Funktionalitäten werden bei Bedarf in der Implementierungsphase entwickelt.

Bei den Herstellern der meisten kommerziellen CMS sind die Programmierer des Systems auch Mitarbeiter des Unternehmens. Das CMS an sich ist bei manchen Anbietern kostenlos, es entstehen nur Implementierungskosten. Andere Anbieter verlangen die erste Zahlung, wenn dem Unternehmen das noch nicht angepasste CMS zur Verfügung gestellt wird. Das heißt die Anfangskosten müssen neben den Implementierungskosten auch mit in die CMS-Anschaffungskosten einkalkuliert werden.

Bei OS-CMS kann das Unternehmen den Entwickler am Markt frei wählen. Durch die freie Verfügbarkeit des OS-CMS entstehen für das CMS an sich keine eigenen Kosten. Allerdings können die Kosten für das Erweitern des Systems je nach Anforderungen und Dienstleister stark schwanken. Deswegen sollte die Suchphase weder zeitlich noch finanziell unterschätzt werden. Unbedingt in die Kalkulation mit aufgenommen werden sollten die Schulungskosten. Gerade bei OS-CMS-Projekten wird hier oft am falschen Ende gespart. Die Effizienz im CMS-Einsatz sinkt deutlich, wenn Redakteure und Administratoren nicht ausreichend geschult sind.

2.2 Lock-In und Unabhängigkeit

Wechselkosten entstehen, wenn ein Kunde den bisherigen Produktanbieter, Dienstleister oder das bisherige Produkt wechselt (Hess und Anding 2003). Diese Kosten entstehen unabhängig davon, ob ein kommerzielles oder OS-CMS verwendet wird. Allerdings gibt es deutliche Unterschiede in der Höhe der Wechselkosten.

Sehen wir uns als erstes Szenario den freiwilligen Wechsel des Produktanbieters bzw. Dienstleisters an: Bei einem kommerziellen Produkt ist die Abhängigkeit von dem bereitstellenden Unternehmen hoch. Das Know-how liegt nur bei dem Unternehmen, das das System entwickelt hat. Oft liegt der Quelltext nicht offen und Informationen über die Struktur der Daten sind schwer zu gewinnen. Dies erhöht die Wechselkosten und erzwingt meist einen Wechsel auf ein anderes System.

Bei einem Open-Source-CMS liegt das Wissen über das System in der jeweiligen Community offen. Je nach Marktanteil und Größe des Projekts gibt es u. U. sehr viele Anbieter, die das System erweitern und administrieren können. Das heißt auch, dass mit einem Dienstleisterwechsel nicht sofort ein Produktwechsel verbunden ist.

Nun zum zweiten Szenario, dem erzwungenen Produktwechsel: Es gibt in der Projektstruktur von Open-Source-Projekten unkalkulierbare Risiken. Beispielsweise könnten sich die wichtigsten Entwickler streiten und die Entwicklung des Systems einstellen bzw., wie bei Mambo geschehen, das Projekt aufspalten. Dies führt zu größeren Unsicherheiten; beispielsweise war einige Zeit unklar, ob man weiter bei Mambo bleiben oder auf das abgespaltene Projekt Joomla! setzen sollte. Diese Risiken sind in etwa vergleichbar mit den Risiken des Konkurses eines kommerziellen CMS-Anbieters. Allerdings kann in einem solchen Fall die Wissensgewinnung über die

für den Umzug notwendigen Strukturen unter Umständen ein zusätzliches Problem darstellen.

2.3 Erweiterbarkeit

Webseiten leben von neuen Funktionen. Im Zuge der durchaus teilweise übertriebenen Web-2.0-Diskussionen haben beispielsweise neue Bedienkonzepte Einzug gehalten, die sich auch in CMS-gestützten Webseiten wiederfinden müssen. In einigen Branchen ist auch ein regelmäßiger *Relaunch* einer Website notwendig. In der Modebranche bringt beispielsweise jede Kollektion eine grundlegende Änderung, in der Industrie reicht ein Relaunch meist alle drei bis fünf Jahre.

Bei Relaunch und Erweiterung um neue Funktionen handelt es sich meist um ein neues Projekt. Allerdings ist hier für Aufwand und mögliche Verbesserungen entscheidend, wie sich das gewählte CMS selbst weiterentwickelt hat. Bei kommerziellen CMS wird diese Entwicklung vom Anbieter getrieben. Dementsprechend sind Erweiterungen natürlich mit Kosten verbunden, die entweder in regelmäßigen Lizenzkosten enthalten sind oder pro neuem Funktionsmodul fällig werden. Bei OS-CMS werden Erweiterungen von der Community entwickelt, müssen allerdings an das eigene System angepasst werden.

In der Praxis besteht ein Hauptproblem sowohl bei kommerziellen als auch bei Open-Source-Erweiterungen darin, dass Funktionalitäten nicht vorliegen bzw. nicht für mehrere Kunden, sondern individuell entwickelt werden. In diesem Fall steigen die Entwicklungskosten stark an. Dementsprechend wichtig ist es, in der Entscheidungsphase für ein System zu wissen, ob es ausreichend schnell weiterentwickelt wird. Da ein solcher Blick in die Kristallkugel in der Regel nicht möglich ist, sollte in der Entscheidungsphase auf jeden Fall die Innovationskraft der Community oder des kommerziellen Anbieters über die letzten Jahre in Betracht gezogen werden.

2.4 Sicherheit und Wartung

Wer sich mit Websicherheit ausführlich beschäftigt, weiß, dass vollständige Sicherheit nicht zu gewährleisten ist. Das liegt vor allem an der weltweiten Zugänglichkeit einer Webseite. Allerdings sind Sicherheitslücken recht gut zu kategorisieren. Die meisten Sicherheitslücken werden nicht durch die Serverkonfiguration oder gesammelte Angriffe wie eine Denial-of-Service-Attacke hervorgerufen, sondern durch Programmierfehler,[7] das heißt also durch Fehler, die bei der Erstellung des CMS selbst gemacht wurden.

Lange Zeit kursierte die Mär, dass Open-Source-CMS generell unsicherer programmiert sind als kommerzielle CMS. Machte man sich diese Behauptung zu eigen, müsste man allerdings davon ausgehen, dass Programmierer besser arbeiten, wenn sie

7 Einen Überblick über die wichtigsten Sicherheitslücken gibt die OWASP-Top-10: http://www.owasp.org/index.php/OWASP_Top_Ten_Project [5. Jan. 2008].

bezahlt werden. In der Praxis entscheidend ist also nicht die Ausrichtung des Projekts oder Produkts, sondern, ob bei den Entwicklern jeweils ein Verständnis für sichere Entwicklung vorhanden ist.

Eine Besonderheit haben Open-Source-CMS allerdings gegenüber kommerziellen Anbietern: Sicherheitslücken werden dokumentiert und dann mittels eines *Patch* geschlossen. Das heißt, dass nicht nur der Finder, sondern jeder versierte Nutzer Informationen über die jeweilige Lücke erhält. Er kann nun die Lücke bei Webseiten austesten, die den entsprechenden Patch nicht eingespielt haben. Im Extremfall findet sich sogar ein böswilliger Nutzer, der ein automatisiertes Skript schreibt, das nach noch nicht gepatchten Webseiten sucht und diese angreift. Die Offenlegung von Sicherheitslücken hat dementsprechend einen großen Nachteil für Unternehmen und Webseitenbetreiber: Sie müssen das OS-CMS regelmäßig aktualisieren bzw. mit ihrem Dienstleister einen entsprechenden Wartungsvertrag schließen. Für das Open–Source-Projekt an sich ist die Offenlegung allerdings dennoch von Vorteil, denn so wird die Qualität des Quelltextes regelmäßig überwacht und verbessert.

Bei kommerziellen Systemen werden gemeldete Lücken meist ohne Kommunikation nach außen oder manchmal auch ohne Kommunikation mit dem Kunden geschlossen bzw. in die normalen Updatezyklen integriert.

2.5 Performance und Skalierbarkeit

High-Performance-Anwendungen sind aktuell eine Domäne von kommerziellen CMS. Ein Grund dafür ist, dass sich Open-Source-CMS generell eher an eine möglichst große Zielgruppe wenden und seltener auf spezielle Anforderungen ausgerichtet sind. Die Entwicklung der letzten Jahre zeigt allerdings, dass einige Open-Source-CMS, wie beispielsweise Typo3 oder Plone, sich durchaus den High-Performance-Bereich erschließen und auch in Mehrserverumgebungen bestehen. Typo3 lässt sich beispielsweise in einer Umgebung mit Lastenausgleich über *load balancer* betreiben (Scholl 2007).

3 Systementscheidung – jedes System ist individuell

Auf den ersten Blick ist die Inhaltsverwaltung eine simple Aufgabe: Die Inhalte sind in der Datenbank gespeichert, über das Backend, die Administrationsoberfläche, lassen sich die Inhalte verwalten und das *Frontend* stellt die Inhalte dar. In der Realität lassen sich diese einfachen Aufgaben aber auf sehr unterschiedliche Arten lösen. Einige CMS organisieren die Inhalte in einer Baumstruktur als Seiten (z. B. Typo3). Einzelne Inhaltselemente sind dann den Seiten zugeordnet. Um Inhalte aus einem Menü auszuschließen oder ganz bestimmte einzuschließen, gibt es mehrere Wege. Redakteure können einzelne Inhalte beispielsweise im Menü verstecken. Außerdem erlaubt z. B. Typo3 mit der eigenen Konfigurations- und Templatesprache *TypoScript*, spezielle Menüs zu erstellen.

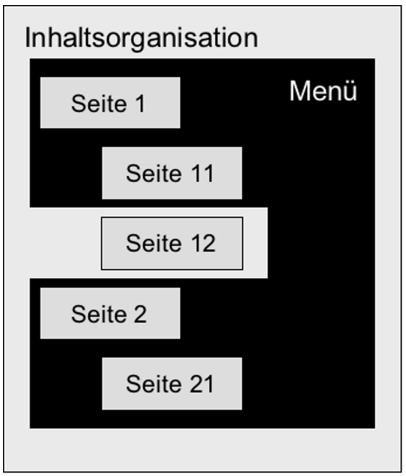

Abbildung 1: Die Inhaltsorganisation in Typo3 – Seite 12 ist aus dem Menü ausgenommen.

Andere Systeme entkoppeln den Inhalt von der Navigationshierarchie (z. B. Joomla!). In Joomla! werden Inhalte in Bereiche und Kategorien einsortiert. Das heißt jeder Inhalt ist eindeutig einem Bereich und einer Kategorie zugeordnet (einzige Ausnahme sind wenige Inhalte ganz ohne Bereichs- und Kategoriezuordnung). Das Menü wird separat erstellt und dort können sowohl ein Bereich als auch eine Kategorie oder ein Beitrag selbst als Menüeintrag definiert werden. Die Darstellung der verschiedenen Menülinks unterscheidet sich dementsprechend voneinander. Eine Kategorieübersicht ordnet die verschiedenen Beiträge beispielsweise als Nachrichtenüberblick an, während ein Beitrag als eigenständige Seite angezeigt wird.

Auch bei der Trennung von Inhalt und Layout gibt es deutliche Unterschiede. Alle CMS verwenden grundlegend ein *Template-System* mit Platzhaltern. Allerdings sind sie in manchen Systemen *XML*-basiert, in anderen in der serverseitigen Technologie geschrieben und wieder andere verwenden eigene Sprachen (vgl. Rawolle 2002, S. 48 ff.).

Aus den verschiedenen Ansätzen ergeben sich natürlich auch unterschiedliche Fähigkeiten, Vor- und Nachteile. Glänzt Typo3 z. B. mit einem sehr umfangreichen Rechtemanagementsystem, so ist Joomla! beispielsweise für Einsteiger wesentlich einfacher zu bedienen. Wer mehrere Webseiten in einem System verwalten möchte, ist dagegen bei Typo3 oder DotNetNuke besser aufgehoben als bei Joomla! und Mambo. Diese konzeptionellen Unterschiede sind dabei nicht nur innerhalb der Open-Source-CMS, sondern natürlich auch innerhalb der kommerziellen Alternativen zu beobachten. Hier hilft vor der Systementscheidung nur eine ausführliche Analyse.

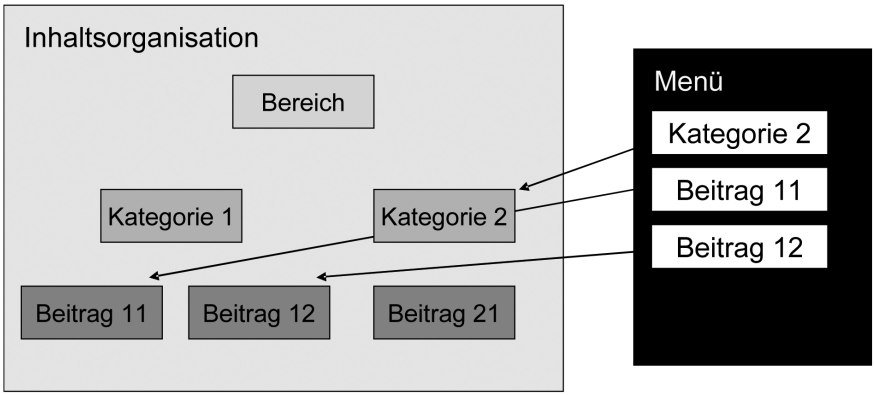

Abbildung 2: Die Inhaltsorganisation von Joomla!

In unserer Beratungspraxis haben sich dazu folgende Schritte bewährt:

1. Anforderungssammlung
2. Kategorisierung und Priorisierung der Anforderungen
3. Systemevaluierung
4. Systemtest mit beteiligten Redakteuren/Administratoren
5. Zusammenfassung der Entscheidungsparameter
6. Systementscheidung

Gerade die Schritte 1, 2 und 3 erfordern von Seiten des Unternehmens, der Agentur und des CMS-Beraters Einarbeitung, Sachkenntnis und Erfahrung. Die hier investierte Zeit kann bis zu 10 Prozent des gesamten Projektvolumens umfassen. Die Abgrenzung von Open Source gegenüber kommerziellen Systemen ist dabei in den Phasen 1, 2, 3 und 5 zu berücksichtigen.

3.1 Technologieentscheidung

Ein nicht zu vernachlässigender Punkt bei der CMS-Wahl ist die *Technologieentscheidung*. Setzt ein Unternehmen beispielsweise auf *ASP.NET* als serverseitige Technologie und eine *Microsoft-Umgebung*, ist der Einsatz eines *PHP-basierten* CMS schwierig bzw. sorgt für einen Technologiebruch. Dementsprechend reduziert sich die Auswahl der verfügbaren Open-Source-CMS. Im ASP.NET-Bereich ist beispielsweise DotNetNuke das am weitesten verbreitete System, kann aber in Sachen Verbreitung und Community-Unterstützung nicht mit PHP-Boliden wie Typo3 oder Joomla! mithalten.

Auch wenn es im Unternehmen keine definierte serverseitige Technologie gibt, weil bisher noch keine Webanwendungen im Einsatz sind, spielt die Technologieentscheidung eine Rolle. Setzt ein Unternehmen beispielsweise auf Plone, ein auf der serverseitigen Technologie *Python* basierendes System, sind zur umfangreichen Weiterentwicklung Python-Fachkräfte notwendig. Diese sind je nach Marktlage deutlich schwerer zu finden.

4 Fazit

Bei der Wahl eines CMS spielt die Entscheidung für oder gegen Open Source auf jeden Fall eine große Rolle. Sie ist aber bei weitem nicht das einzige Kriterium. Im Vordergrund sollten immer eine Anforderungsanalyse und vorausschauende Planung stehen. Die Entscheidung ist vor diesem Hintergrund strategischer und individueller Natur, und ihre Bedeutung steigt mit der Bedeutung der Webseite, des Extra- oder Intranets, das mit dem CMS verwaltet wird.

Unterschätzt wird hier oft die Langfristigkeit einer Entscheidung. Im Idealfall begleitet ein CMS ein Unternehmen über viele Jahre und wird ständig gewartet und weiterentwickelt. Dies wird oft bei der Projektplanung zu Gunsten der schnellen Entscheidung und kurzfristigen Kostenreduzierung vernachlässigt. Ist das Marketing für die Webseite zuständig, sind es oft einjährige Planungszyklen mit entsprechend beschränktem Budget. Und auch in der IT-Abteilung sind viele Systeme und vor allem Hardwarekomponenten wesentlich kurzlebiger als ein CMS.

Literatur

Baumgarten, U., Krcmar, H., Reichwald, R. und Schlichter, J. (2001), Community Online Services and Mobile Solutions – Projektstartbericht des Verbundvorhabens COSMOS, techn. Bericht, Technische Universität München.

Hess, T. und Anding, M. (2003), Wechselkosten und Lock-in-Effekte bei Medienprodukten, *in* G. Brösel (Hrsg.), 'Medienmanagement: Aufgaben und Lösungen', Oldenbourg, München, S. 85–94.

Rawolle, J. (2002), *XML als Basistechnologie für das Content Management integrierter Medienprodukte*, Deutscher Universitätsverlag, Wiesbaden.

Scholl, M. (2007), 'Lastenausgleich mit TYPO3', *t3n magazin* **7**. http://t3n.yeebase.com/magazin/ausgaben/artikel/lastenausgleich-mit-typo3/.

Change Management: Linux-Desktop-Migration mit Erfolg

BEATE GROSCHUPF UND NATASCHA ZORN

Desktop-Migrationen scheitern oftmals an Widerständen von Anwendern und Administratoren. *Change Management* wird als eine Methode vorgestellt, die diese hartnäckig verbreiteten Widerstände abbauen und so dem Erfolg von Migrationsprojekten den Weg frei machen kann. Die Autorinnen erläutern in dem Artikel Hintergründe und Ursachen, stellen die von ihnen ausgearbeitete fünfstufige Methode dar und zeigen anschließend am Praxisbeispiel des zurzeit weltweit größten Desktop-Migrationsprojekts des Auswärtigen Amtes den Nutzen und die Umsetzung der Methode.

Schlüsselwörter: Migration · Benutzer · Migrationskonzepte · Linux · Change Management · OpenOffice.org

1 Wie oft wechseln denn Sie Ihren Lieblingsitaliener?

Eigentlich nie, sonst wäre er ja nicht der Lieblingsitaliener. Und warum sollten Sie auch. Bei ihm wissen Sie, woran Sie sind. Sie wissen, wo man am besten parkt, Sie werden persönlich begrüßt, Sie bekommen den schönsten Tisch, treffen auf bekannte Gesichter, werden bevorzugt bedient, kennen die beste Pizza und auch bei neuen Gerichten sind Sie doch vor unangenehmen Überraschungen sicher. Da weiß man, was man hat.

Ähnlich verhalten Sie sich, wenn Sie zum Beispiel einen neuen Computer kaufen und sich für ein Betriebssystem entscheiden müssen. Der Endanwender, so viel er auch darüber schimpfen mag, benutzt im Normalfall die Programme, die er kennt. Die, die üblich sind, die, die seine Freunde auch benutzen. Sonst müsste man etwas Neues lernen, dann ist das neue System womöglich nicht kompatibel mit Systemen von Kunden und Partnern, dann verliert man vielleicht auch noch an persönlichem Marktwert, weil man das alte *MS-Office* nicht mehr perfekt beherrscht und überhaupt

haben doch alternative Systeme, wie Linux, sicherlich nicht alle gewohnten Funktionen und sind nur halb so gut und das Lernen kostet Zeit, dann gehen Daten verloren und man muss sich vor dem Chef rechtfertigen. Die Liste der typischen Einwände ließe sich ins beinahe Endlose fortsetzen.

Dieses typische Verhalten von Endanwendern stellt auch Entscheidungsträger bei Linux-Migrationen in Unternehmen und Behörden vor Probleme. Daher wird Linux oft nur in Backendsystemen eingesetzt, obwohl Kosteneinsparungen auch im Frontend-Bereich, auf den Arbeitsplatzrechnern der Mitarbeiter, möglich wären, was Freiräume für Investitionen in Innovation schaffen könnte. Dennoch wird eine solche Migration nur sehr selten umgesetzt. Wo liegen hierfür die Gründe? Und welche Schritte und Vorgehensweisen sind notwendig, um für Linux auf dem Desktop bei Entscheidern und Anwendern gleichermaßen eine ganzheitliche und dauerhafte Akzeptanz zu schaffen?

Wie viele Studien[1] belegen, ist der Faktor Mensch der Hauptgrund für das Scheitern großer Implementierungsprojekte. Insbesondere Vorhaben, bei denen sich für die Endanwender Änderungen im Arbeitsalltag ergeben, werden häufig durch Widerstände der Anwender verzögert oder durch Boykott als Ganzes gefährdet. Kommunikationsschwierigkeiten zwischen IT und Business sowie mangelnde Unterstützung durch das Management sind weitere Probleme, die auf den Faktor Mensch zurückzuführen sind – und das unabhängig vom eingesetzten Betriebssystem und den darauf basierenden Anwendungen.

Die Einführung neuer Systeme, Umstrukturierungen, globale Projekte oder neue Formen der Arbeitsorganisation, um nur einige zu nennen, können Veränderungen sein, deren Bewältigung ein spezielles Vorgehen bei der Umsetzung erfordern: *Change Management*, das sich der Dynamik, dem Managen und Führen von Veränderungsprozessen annimmt. Diese Methode hat ihre Ursprünge in den fünfziger Jahren des 20. Jahrhunderts. Seit damals entwickelte sie sich über Zwischenstationen, wie den Bereichen der strukturellen bzw. strategischen Veränderungen und mitarbeiterzentrierten Ansätzen, hin zu einer ganzheitlichen Betrachtungsweise des Themas *Veränderungen steuern*.

Change Management wird seitdem als Teilbereich des Projektmanagements mit dem Ziel eingesetzt, alle von einem Veränderungsprozess Betroffenen gezielt so früh wie möglich einzubeziehen, um so den Projekterfolg zu sichern. Im Zusammenhang mit Linux-Migrationen und bei der Einführung von Open-Source-Anwendungen stand die Methode *Change Management* bisher nicht im Vordergrund, weil Linux fast ausschließlich im Serverbereich, also für den Endbenutzer am Arbeitsplatz „verborgen", eingesetzt wurde. Dies ändert sich nun. Die Ursache hierfür lässt sich benennen: Linux auf dem Desktop ist ein Thema, das in jüngerer Zeit breites Interesse und Ak-

1 Siehe „Statistics over IT projects failure rate" auf den Seiten von IT-Cortex (http://www.it-cortex.com/Stat_Failure_Rate.htm) und „Helping Employees to embrace change" auf McKinsey Quarterly (http://www.mckinseyquarterly.com/article_abstract_visitor.aspx?ar=1225&L2=18&L3=27) [16. Nov. 2007].

zeptanz findet, nicht zuletzt, weil der Einsatz herstellerunabhängiger Software und die damit verbundene Kostenersparnis für viele Unternehmen zunehmend interessant werden. Wie *Change Management* den Prozess erfolgreicher Migrationen unterstützt und warum die Methode die anfängliche Ablehnung von Anwendern erfolgreich aufnimmt und positiv verändert, soll in den folgenden Abschnitten dargestellt werden.

2 Widerstände und ihre Gründe

Widerstände sind in der Arbeitswelt ein ganz alltägliches Phänomen und eine normale Begleiterscheinung jedes Entwicklungsprozesses. Es gibt in der Praxis kein Lernen und keine Veränderung ohne Widerstand. Doch wo immer ein Widerstand auftaucht zwingt er zu Denkpausen, zu klärenden Gesprächen, vereinzelt sogar zu Kurskorrekturen. Konstruktiver Umgang mit Widerstand ist deshalb einer der zentralen Erfolgsfaktoren beim Management von Veränderungen. Die am häufigsten anzutreffenden Widerstände sind nach Doppler und Lauterburg (1999):

1. Die Anwender haben die Ziele, Hintergründe oder Motive der Veränderung *nicht verstanden*.

2. Die Anwender haben zwar verstanden, warum die Veränderung notwendig ist, sie sind aber *nicht von ihrer Notwendigkeit überzeugt*.

3. Die Anwender haben die Beweggründe verstanden und sind auch von der Notwendigkeit überzeugt, aber sie wollen oder vermögen die Veränderung nicht mitgehen, weil sie sich davon *keine persönlich positiven Konsequenzen* versprechen.

Die häufigste und zugleich die am schwierigsten zu lösende Form von Widerstand liegt im letztgenannten Fall vor, wenn sich nämlich die Betroffenen von den vorgesehenen Maßnahmen keine positiven Konsequenzen versprechen.

Offensichtliche Gründe für die Ablehnung von derartigen Prozessen, wie z. B. einer Linuxmigration, sind eine natürliche Angst und verbreitete Vorurteile gegenüber neuen Technologien, oftmals allem Neuen. So hat Leon Festinger schon 1957 in seiner Theorie zur kognitiven Dissonanz, dem *inneren Widerspruch*, festgestellt, dass Stress entsteht, wenn neue Informationen nicht zu den eigenen bisherigen Verhaltensweisen passen (Festinger et al. 1978). Es entsteht eine innere Abwehrhaltung dem Neuen gegenüber, die Reaktion ist Widerstand in passiver oder aktiver Form. Anders ausgedrückt: Einmal getroffene Entscheidungen werden zunächst und soweit wie möglich beibehalten, auch wenn neu erworbene Informationen objektiv dagegen sprechen. Speziell bei Linux und anderen Open-Source-Projekten ergeben sich zusätzlich Widerstände, die aus gängigen Vorurteilen gegenüber *F/OSS* herrühren.

Ängste und Vorurteile der Anwender dürfen im gesamten Verfahren nicht unterschätzt oder ignoriert werden. Untrainierte, lustlose oder gar bewusst die Umstellung

ablehnende Benutzer bergen für das Unternehmen direkt (z. B. für zusätzliche Trainings) und indirekt (wie durch ineffizientere Arbeitsabläufe) einen signifikanten Kostenaufwand. Unwille, Demotivation oder im schlimmsten Fall ein Boykott gefährden den Erfolg der Investition und damit das Projekt als Ganzes. Der langfristige Nutzen der Investition und der durchgängige Gebrauch der neuen Systeme können jedoch mit Hilfe von professionellem *Change Management* gesichert werden.

3 Schritt für Schritt zum Wechsel

Was ist also zu tun, um einen gut motivierten Umstieg und die durchgängige Nutzung z. B. von Linux sicherzustellen? Was müsste passieren, damit Sie einem neuen Lieblingsitaliener nicht nur eine Chance geben, sondern auch tatsächlich langfristig wechseln? Den Fall, dass Ihr alter Italiener seine Türen für immer schließt, möchten wir einmal ausschließen. Aber was wäre, wenn Sie erführen, dass ein neuer Italiener das gleiche Flair, die gleiche oder gar bessere Qualität bei niedrigeren Preisen anbieten würde? Vielleicht wären Sie skeptisch, aber Sie würden ihm doch wahrscheinlich eine Chance geben! Doch bei diesem Ausprobieren darf nichts schief gehen, denn sonst bleiben Sie bei Ihrem alten Italiener. So gilt auch für Migrationsprojekte als Maxime: Die verbreiteten Vorurteile dürfen nicht wahr werden – die Migration muss für die Nutzer aktiv gesteuert werden! Das neue System soll sich den Benutzern von seiner besten Seite zeigen können.

Betrachten wir Möglichkeiten und Lösungen, die *Change Management* bietet. In einem erfolgreichen Migrationsprojekt durchlaufen die zukünftigen Nutzer eines neuen Systems fünf Schritte. Um diese zu steuern, können sechs Maßnahmen eingesetzt werden.

Die erste Maßnahme, *überzeugen*, ist die Hauptstellschraube, sie ist ursächlich für den Projekterfolg. Trotzdem wird sie oft übergangen. Ziel dieser ersten Maßnahme ist es, die Betroffenen in die Phasen *verstehen* und *wollen* zu führen, um ihre ursprüngliche und ganz natürliche Abwehrhaltung in ein erfreutes Annehmen des Neuen zu wandeln.

Zunächst muss also die Motivation der Anwender und Entscheidungsträger sichergestellt werden. Essentiell ist hierfür, dass die Notwendigkeit und die Vorteile der Migration von den betroffenen Führungskräften und Mitarbeitern verstanden und gewollt werden. Dazu werden die der Entscheidung zugrunde liegenden Hintergründe erklärt und der zu erzielende Nutzen sowohl für das Unternehmen als auch für jeden Einzelnen aufgezeigt. Eine *attraktive Zukunftsvision* muss entstehen. Eine intensive Kommunikation ist Teil dieser Maßnahme. In Workshops werden Vorurteile aufgenommen und umgewandelt. In der internen Kommunikation wird im Detail über das System und seine Vorteile berichtet. Um das bekannte Sprichwort einmal umzuformulieren: Es genügt nicht, zu kommunizieren, dass ein neues Boot gebaut werden soll, sondern welche fernen, schönen Länder damit bereist werden können.

Des Weiteren sollte in dieser Phase das Phänomen *not invented here* beachtet werden. Eine Untersuchung von Katz und Allen (1982) deutet darauf hin, dass Menschen Dinge, die nicht von ihnen selbst erarbeitet, gefunden, erfunden usw. wurden, allein aus dem Grund der persönlichen Nichtbeteiligung intuitiv ablehnen. Ursachen hierfür sind fehlendes Verständnis und mangelnde Identifikation mit der Neuerung. Eine Idee, die Sie selbst erarbeitet haben, vertreten Sie gegenüber anderen leichter und schneller als eine Idee, an der Sie nicht beteiligt waren. Die Identifikation kann durch ein möglichst frühes Einbinden von Nutzern in das Projekt herbeigeführt werden. So sind Projektteams immer auch mit Mitgliedern der fachlichen Seite und über Funktionen und Bereichsgrenzen hinweg zu besetzen. Diese Mitglieder sollten an Projektplanung und -durchführung aktiv teilnehmen. Durch das bewusste Einbinden von Meinungsmachern wird zusätzlich ein positives Projektimage erzeugt.

Nach dieser ersten Phase ist der schwierigste, der menschliche Teil geschafft. Die folgenden Schritte sind von gleicher Wichtigkeit, aber direkt beeinflussbar und regelbar und dadurch einfacher.

Die nächste Maßnahme *veröffentlichen* stellt sicher, dass die neuen Systeme inklusive aller notwendigen Zugangsdaten, Hilfsmittel und Werkzeuge allen Endbenutzern zur Verfügung gestellt werden. Ein detaillierter Rollout-Plan wird erstellt und stufenweise vom Team umgesetzt. In vielen Fällen wird hierzu ein firmeneigenes Intranet genutzt und mit allen wichtigen Informationen und Werkzeugen versehen. Weiterhin ist mit der darauf folgenden Maßnahme *verankern* zu prüfen, ob sich durch die neuen Systeme organisatorische Änderungen oder Veränderungen der Arbeitsprozesse ergeben müssen. Fragestellungen wie „Gibt es Änderungen in den Teamstrukturen?“ oder „Sind neue Genehmigungsprozesse umzusetzen?“ sind bis zu diesem Zeitpunkt zu klären. Die Ergebnisse dieser Analyse müssen dann umgesetzt werden und in interne Prozesse einfließen. Ziel dieser beiden Maßnahmen ist es, sicherzustellen, dass die Benutzer die Stufe *dürfen* erreichen: Sie haben unkomplizierten, vollständigen Zugang zu den Systemen. Organisatorisch ist sichergestellt, dass alle notwendigen Nutzungs- und Weisungsbefugnisse gegeben sind.

Die nächste Maßnahme *trainieren* findet in den meisten Projekten statt. Oftmals wird von ihr fälschlicherweise angenommen, dass sie als Einzelmaßnahme ausreicht und die vier restlichen Maßnahmen ersetzt. Ergebnis dieser Phase ist jedoch lediglich, dass die Nutzer die Stufe *können* erreichen können. Die Nutzer sind mit dem Erreichen dieser Stufe in der Lage, das System zu nutzen.

Wie bereits aufgezeigt, reicht dieser Vorgang allein nicht aus, um die reibungslose Nutzung des Systems und damit Investitionssicherheit tatsächlich herzustellen. Aufgabe dieser Phase ist es, Trainings zu erstellen, zu planen und durchzuführen, so dass alle Nutzer sicher mit den Systemen umgehen können und die relevanten neuen Funktionalitäten beherrschen. An dieser Stelle ist es wichtig, hierarchisch vorzugehen und mit den Trainings bei der Führungs- oder Leitungsebene zu beginnen, um den Vorbildeffekt für das Projekt gezielt zu nutzen.

Nun ist die Organisation im Prinzip startklar. Aber auch der Start muss effektiv für den Projekterfolg genutzt werden. So ist es Ziel der Maßnahme *starten*, dass die Nutzer die nächste Stufe *müssen* erklimmen. Ohne ein festes Startdatum verliert das Projekt an Wirkung und Ernsthaftigkeit. Fehlende Ernsthaftigkeit wiederum gefährdet den *return on investment*, der nur durch hohe, stabile Nutzungsraten sichergestellt wird. Also sind ein festes Startdatum, Ansprechpartner für Fragen und auch FAQ zu kommunizieren. Der Starttag wird positiv vermarktet, beispielsweise mit begleitenden Events in der Kantine, Postern in Gängen oder anderen Mitteln. Je professioneller dieser Schritt gestaltet wird, desto positiver wird das Projekt wahrgenommen, was direkten Einfluss auf den Projekterfolg hat. Auch dieser Schritt wird oft unterschätzt und ausgelassen.

Zu einem erfolgreichen Projektabschluss gehört letztendlich auch die Maßnahme *nachhalten*, nämlich die Durchführung je eines Reviews nach zwei Wochen und nach sechs Monaten. Bei diesen Reviews werden die Nutzer, die Sponsoren des Projekts und das Projektteam befragt, sich ergebende Schwierigkeiten erfasst und gelöst, *lessons learnt* für zukünftige Projekte festgehalten und positive Ergebnisse intern vermarktet.

Professionell geplantes und stringent durchgeführtes *Change Management*, das die Projektleitung in einem Migrationsprojekt unterstützt, hilft, die Nutzer von einem neuen System und seinen Vorteilen zu überzeugen, sie zu einer sicheren und vollständigen Nutzung zu befähigen und ein positives Projektimage zu generieren. *Change Management* ist eine Investition, die die Gesamtinvestition sichert und Projekte zum Erfolg führt. Mit *Change Management* können Sie Ergebnisse erzielen und feiern – vielleicht bei Ihrem neuen Lieblingsitaliener.

4 Change Management in der Praxis

Im Rahmen der Linux-Migrationsstrategie des Auswärtigen Amtes (vgl. Werner 2007) stand im Frühsommer 2007 erstmalig die Einführung von Linux auf Desktoprechnern einer Niederlassung des Auswärtigen Amtes an. Bisher wurde zwar Linux schon standardmäßig auf Laptops eingesetzt und war zudem per *dual boot* auf stationären Rechnern verfügbar, nun aber sollte in einem Pilotprojekt der durchgängige Einsatz von Linux in einer ausgewählten Niederlassung erprobt und bei positiven Ergebnissen weltweit umgesetzt werden.

Um die Akzeptanz der Migration durch die Anwender zu prüfen und den reibungslosen Ablauf des Pilotprojekts sicherzustellen, wurde das Team von *CHANGE* beauftragt, die Pilotmigration mit ihrem Veränderungsmanagement zu begleiten. In Zusammenarbeit mit dem IT-Team des Auswärtigen Amtes wurde entlang des *CHANGE–Kreislaufes* das oben dargestellte sechsstufige Vorgehen gewählt und auf die speziellen Rahmenbedingungen des Auswärtigen Dienstes angepasst.

In der ersten Phase *überzeugen* wurde sichergestellt, dass die Anwender frühzeitig von dem Management vor Ort über die Migration, das geplante Vorgehen und die Beglei-

tung durch *CHANGE* informiert wurden. Hierzu wurde ein spezielles Informationsdokument für das Management vorbereitet, das die wichtigsten Aussagen beinhaltete. Weiterhin wurde kurz vor der eigentlichen Migration eine Informationsveranstaltung für alle Anwender durchgeführt, in der Vorgehen, Zeitplan, Verantwortlichkeiten und die Gründe für die Migration vorgestellt, diskutiert und Fragen beantwortet wurden. Um mögliche Widerstände, Ängste und Befürchtungen aufzunehmen und gezielt abbauen zu können, wurde in den Tagen vor der Migration eine umfassende Anwenderumfrage von *CHANGE* organisiert. So wurden mit allen vor Ort anwesenden Benutzern aus allen Hierarchieschichten 20-minütige Einzelinterviews durchgeführt. Ziel war es, die Erwartungen abzufragen, also mögliche Vorurteile, Ängste, das Image von Linux und bisherige Erfahrungen mit Open-Source-Anwendungen, sowie den Status quo der Zufriedenheit festzuhalten. Auch mit den lokalen IT-Verantwortlichen wurden Einzelinterviews geführt, da sich auch für sie Änderungen durch die Umstellung ergeben, deren Umfang oft unterschätzt und deren Durchführung zu wenig gezielt unterstützt wird.

In der zweiten Phase *veröffentlichen* wurde zur Eingewöhnung bereits sechs Wochen vor dem Start der Migration die Standardverknüpfung von *MS-Office* hin zu *OpenOffice.org* umgestellt, sodass die Anwender bereits früh die Möglichkeit hatten, die neue Anwendung kennenzulernen und auszuprobieren. In diese Phase fielen auch die Planung und Durchführung der Datensicherung und schließlich die Migration selbst, die an zwei aufeinander folgenden publikumsverkehrsfreien Tagen, an einem Wochenende, umgesetzt wurde.

Die dritte Phase *verankern* wurde per Erlass sichergestellt, sodass die Migration und der Einsatz der neuen Anwendungen ab dem Starttag bindend für alle Anwender war.

Phase vier, das *Trainieren*, wurde durch das IT-Team der Zentrale verwirklicht. So konnte die Qualifizierung über gezielte Gruppentrainings und durch individuelle Frage- und Betreuungstermine mit den Anwendern sichergestellt werden.

Am eigentlichen Starttag, der Phase fünf, *beginnen*, wurden die Anwender von einer linuxspezifischen Überraschung, einem Spielzeug-Tux („*goodie*“) sowie Trainingsunterlagen und Leitfäden an jedem Arbeitsplatz begrüßt. So konnte die neue Ära mit einem positiven, unterstützenden und spielerischen Signal beginnen. Ferner erfolgte während der ersten beiden Tage eine intensive Anwenderbetreuung vor Ort durch das IT-Team. Damit konnte sichergestellt werden, dass sich die in der Umfrage geäußerten Ängste hinsichtlich Daten- und Kompetenzverlust sowie Arbeitsunfähigkeit nicht realisierten und sich die Anwender hervorragend betreut fühlten. Im Ergebnis war das gesamte Botschaftsteam von Tag eins an arbeitsfähig, kleinere Probleme konnten rasch und erfolgreich behoben werden.

Auch die letzte Phase, *nachhalten*, wurde vor Ort eingeleitet. In dieser Phase führte das Team von *CHANGE* Post-Ex-Interviews mit den Anwendern. Ziel dieser Umfrage war es, die tatsächlichen Erfahrungen und die Zufriedenheit der Nutzer aufzunehmen, sie mit den vorher geäußerten Befürchtungen zu vergleichen und von

den Anwendern selbst Optimierungsvorschläge, Kritik und Wünsche für künftige Migrationen aufzunehmen. Hierbei wurden insbesondere die persönliche Einschätzung zum Migrationserfolg und zu den neuen Systemen sowie Probleme, entstandener Mehr- oder Minderaufwand und Optimierungsvorschläge für zukünftige Migrationen abgefragt. Neben *CHANGE* war auch ein Vertreter der zentralen IT mit vor Ort, der für letzte Anwenderprobleme und -fragen zur Verfügung stand.

Die Befragung vor der Migration ergab, dass bei knapp 60 Prozent der Anwender Widerstände und Ängste vorhanden waren. Diese basierten zu gleichen Teilen auf Vorurteilen sowie auf eigenen negativen Erfahrungen mit Open Source-Anwendungen. Thematisch waren mangelnde Kompatibilität von Dateien und Datenverlust durch die Migration die Hauptängste. Der damit befürchtete Mehraufwand, Rechtfertigungsdruck und Kompetenzverlust waren die dahinterliegenden Ursachen.

Nach der Migration konnte ein komplexes, z. T. widersprüchliches Bild festgestellt werden. Zum einen erlebten rund 50 Prozent der Befragten einen anhaltenden Minderaufwand durch die Umstellung und befürworteten grundsätzlich die Migration als solche. Gleichwohl erlebten 80 Prozent der Befragten die Umstellung und die neuen Anwendungen als problematisch, zeitaufwändig und ohne klaren, persönlichen Mehrwert. Daraus kann geschlossen werden, dass bei ca. 30 Prozent der Befragten der persönliche Unmut gegenüber der Migration trotz des erlebten Minderaufwands überwiegt.

In der Analyse wurde zum einen festgehalten, dass dies ein durchschnittliches, nicht überraschendes Ergebnis darstellt, denn solche Befragungen beinhalten erfahrungsgemäß immer auch einen Anteil an allgemeiner Unzufriedenheit, die unabhängig von der anstehenden Veränderung ist. Unter Berücksichtigung dieser Tendenz konnte dennoch festgestellt werden, dass die Migrationsstrategie zwar von den Anwendern grundsätzlich befürwortet wurde, die Umsetzung in Bezug auf Anwenderzufriedenheit jedoch Optimierungspotenzial birgt. Die 60 Prozent der Skeptiker enthalten ein oben prognostiziertes Boykottpotenzial, das unter Sicherheitsaspekten äußerst problematische Auswirkungen haben kann.

Die Ergebnisse der beiden Befragungen und die Ergebnisse einer teaminternen Lessons-learned-Sitzung wurden zum Abschluss des Pilotprojektes aufbereitet, analysiert und mit entsprechenden Empfehlungen für die weltweite Migration versehen.

5 Ausblick

Die in den Interviews aufgenommenen konkreten Probleme und Befürchtungen in der Pilotvertretung selbst wurden mit individuellen Maßnahmen vom Projektteam direkt im Anschluss an die Befragung vor Ort gelöst.

Für den anstehenden weltweiten Rollout wurden von *CHANGE* folgende Maßnahmen für die Themen Image, Erwartungen und Kompetenzen abgeleitet. Dabei

wurden als Rahmenbedingungen geringer (personeller) Aufwand, Skalierbarkeit sowie ein hoher Standardisierungsgrad berücksichtigt:

Image Eine amtsweite, den Rollout begleitende Imagekampagne, die die Realität und Vorurteile gegenüber Linux und Open Source klar darlegt und ein positives Image schafft.

Erwartungen Ein Kommunikationspaket für jede zu migrierende Vertretung, die Zielsetzung, Nutzen, Aufwand, Ablauf und Aufgaben für alle Anwender kurz und positiv umfasst.

Kompetenzen Ein webbasiertes Training, das die weltweit verteilten Anwender vor der Migration auf die neuen Anwendungen vorbereitet, sodass negative eigene Erfahrungen und der damit verbundene Umstellungsaufwand und Frust minimiert werden können.

Mit Hilfe dieser Möglichkeiten kann der weltweite Rollout und die Umsetzung der IT-Strategie dank kompetenter, informierter und zufriedener Anwender reibungslos fortgeführt werden. Erfolgreich umgestellt wurden in 2007 die Botschaften in Oslo, Kairo, Eriwan, Madrid, Daressalam und das Generalkonsulat in St. Petersburg. Die Migration wird weltweit kontinuierlich fortgesetzt.

6 Fazit

Innovativ und einmalig war in diesem Projekt der Einsatz von *Change Management* in einem Open-Source-Migrationsprojekt. Hierdurch konnte mit modernen Managementmethoden ein Projekt, das in diesem Umfang weltweit ohne Vergleich ist, unterstützt und zum Erfolg geführt werden. Durch die Betrachtung nicht nur technischer, sondern auch psychologischer Parameter konnte die Qualität der Umsetzung gesteigert und die Effektivität der Ressourcennutzung verbessert werden.

Die Ergebnisse des Projekts werden in die kommenden Migrationsprojekte des Auswärtigen Amtes einfließen und garantieren so eine kostengünstige und schnelle Umsetzung von Migrationen bei gleichzeitiger Steigerung der Zufriedenheit der Mitarbeiter.

Zusammenfassend lässt sich festhalten: Nicht nur die Theorie, sondern auch die Praxis z. B. der erfolgreichen Migrationen des Auswärtigen Amtes zeigen, dass *Change Management* eine wirksame Methode darstellt, Linux-Desktop-Migrationen erfolgreich zu unterstützen und umzusetzen.

Durch das gezielte Einbinden der Nutzer kann die vorherrschende Skepsis gegenüber Open-Source-Software aufgefangen, kanalisiert und abgebaut werden. So werden Projekte nicht boykottiert, verlangsamt oder verteuert, sondern rundum unterstützt und gemeinsam mit allen Beteiligten erfolgreich ins Ziel gebracht.

Literatur

Doppler, K. und Lauterburg, C. (1999), *Change Management. Den Unternehmenswandel gestalten*, 11 Aufl., Campus Verlag, Frankfurt/New York.

Festinger, L., Irle, M. und Möntmann, V. (1978), *Theorie der Kognitiven Dissonanz,* 4 Aufl., Huber-Verlag, Göttingen.

Katz, R. und Allen, T. (1982), 'Investigating the Not Invented Here Syndrome', *R&D Management* **12**, S. 7–19.

Werner, T. (2007), World Domination: Die Erfolgsgeschichte der Linux- und Open-Source-Einführung im Auswärtigen Amt, *in* B. Lutterbeck, M. Bärwolff und R. A. Gehring (Hrsg.), 'Open Source Jahrbuch 2007 – Zwischen Freier Software und Gesellschaftsmodell', Lehmanns Media, Berlin, S. 239–248.

Kapitel 5

Vom Wissen zur Vernetzung

„There are three kinds of death in this world. There's heart death, there's brain death, and there's being off the network."

– Guy Almes

Fest im Griff – Netze und Menschen

MATHIAS EBERLE

Das Spannungsfeld von Wissen, Wissensbewahrung und Wissensweitergabe verdient es auch in dieser Ausgabe des Open Source Jahrbuchs, in einem eigenen Kapitel betrachtet zu werden. Die vergangenen Jahre haben starke Auseinandersetzungen zu den Themen Open Access, Open Content und Creative Commons gesehen. In diesem Jahrbuch können wir Beiträge vorstellen, die sich um etwas weniger beachtete, aber nicht weniger wichtige Teilgebiete dieser Debatte kümmern.

Schon früh erkannten Menschen, dass es nur dieses eine gab, was unsere Zivilisationen gegenüber den dunklen Zeitaltern auszeichnete, aus denen schließlich der Mensch und seine frühesten Gesellschaftsformen entstanden: Wissen. Wissen bedeutete den individuellen Erwerb und die Weitergabe von Fähigkeiten und Fertigkeiten, durch deren Einsatz der Mensch seine täglichen Arbeiten vereinfachen konnte. Durch die Beherrschung des Ackerbaus konnten sich frühe Gesellschaften bilden, die nicht länger auf ein anstrengendes und gefährliches Nomadenleben angewiesen waren. Der Mensch wurde sesshaft; die effektivere Verwendung von Ressourcen und ein Bedürfnis nach ideellen Gütern brachten schließlich Kultur hervor, die Malerei, Bildhauerei, Literatur und schließlich Musik umfasste.

War individuelles Wissen gefährdet, wurden Menschen erfinderisch. So wurde schon im Altertum die Schrift entwickelt, die es zum ersten Mal in der Geschichte ermöglichte, das Wissen schon lange verstorbener oder weit entfernt lebender Personen mit verhältnismäßig geringem Informationsverlust weiterzugeben. In Zeiten, in denen einzelne Schriftwerke aus z. B. ideologischen oder praktischen Gründen gefährdet waren, wurden Bibliotheken und Klöster gebaut, die als Inseln des Wissens Informationen über die Zeiten retten konnten.

Wir stellen heute fest, dass dieses tradierte Verständnis von Wissen und Informationen nicht falsch, aber vielleicht etwas eindimensional gedacht ist. Seit es menschliche Gesellschaften gibt, lag das Augenmerk auf den individuellen Fähigkeiten des Einzelnen. Das Verständnis von der Gemeinschaft, die schließlich stark macht, ist zwar schon lange bekannt. Die technischen Fähigkeiten, unsere individuellen intellektuellen

Leistungen an jedem Ort und zu beinahe jeder Zeit zu vereinigen und etwas Größeres als die Summe der jeweiligen Fertigkeiten zu schaffen, sind aber nur ein paar Jahrzehnte jung und entziehen sich teilweise noch unserem Verständnis.

Diese von uns als Netze bezeichneten Einrichtungen üben auf viele ihrer Teilnehmer eine starke Anziehung aus. Sie ermöglichen den Teilhabenden, beinahe unmittelbar Wissen und Kultur auszutauschen und sich gegenseitig zu bereichern, ohne zu verlieren, also gegen jede Zweckrationalität (nach Weber) zu handeln und erreichtes Wissen der Gesellschaft zur Verfügung zu stellen. Dabei werden Mechanismen genutzt, die durchaus im Widerspruch zu eher traditionellen Formen sozialer Interaktion stehen: So wird den Nutzern von Netzen zwar direkte, unmittelbare Kommunikation und Information geboten, der Preis ist aber ein weitgehendes Verzichten auf die Vertraulichkeit früher als persönlich betrachteter Informationen. Diese zunächst theoretischen Betrachtungen finden sich ganz konkret in einigen aktuellen Beispielen.

Genannt sei hier die in diesem Jahr gesehene Debatte um das Studentennetzwerk *StudiVZ*. Die Betreiber des Netzes wollten einen höheren, vielleicht zu hohen *Preis* von den Nutzern, ein (heute) zu hohes Maß an Verfügungsgewalt über als vertraulich empfundene Daten. Derartige Konflikte werden sicher auch in den kommenden Jahren eine Rolle spielen.

Interessant ist dabei, dass diese Auseinandersetzungen inzwischen beinahe ausschließlich selbst über ein Netz, das Internet, ausgetragen wurden, zu einem größeren Teil sogar in eben diesem *StudiVZ*. Die Mechanismen, die unsere heute schon explizit als „sozial" bezeichneten Netze zusammenhalten, scheinen also außergewöhnlich stark zu sein, stärker als die wahrgenommenen sozialen Wirkungen von Aufrufen von Sportvereinen oder Kirchengemeinden. Das ist nur auf den ersten Blick verwunderlich – ist doch die Zahl der Partizipierenden und damit das Maß der womöglich dem Einzelnen erteilten Anerkennung oder sozialen Belohnung potenziell so groß wie die Gesamtzahl aller Partizipierenden aller Netze überhaupt.

Genau so gibt es auch ein Maß der sozialen Bestrafung, das größer und schwerwiegender ist als viele andere: Sanktion durch sozialen Ausschluss, Nicht-Wissen und Nicht-beteiligt-Sein an Information und netzimmanenten Prozessen. Wie groß die Hürde ist, ein solches System wieder zu verlassen, weiß jeder, der einige Zeit mit sozialen Netzen verbracht hat.

Georg Hüttenegger stellt in diesem Kapitel den Begriff „Wissensmanagement" vor und untersucht, mit welchen Open-Source-Produkten und in welchen Kombinationen sich sinnvoll Wissen in Unternehmen verwalten und vermehren lässt. Content–Management-Systeme, Groupware-Produkte und Document-Management-Systeme werden vorgestellt und verglichen.

In seinem Beitrag zu Richard Stallmans „Goldener Regel" und den „Digital Commons" untersucht Glyn Moody Fragen zum Bereich des *sharing* in der Kultur der freien Software. Er stellt Fragen zum Ursprung dieser Kultur des Weitergebens und betrachtet das Konfliktfeld von freier Software und Wirtschaft.

Die Konservierung bestehenden Wissens, Archivierung, steht im Mittelpunkt des Artikels von Christoph Jeggle und Ulrich Kampffmeyer. Sie stellen fest, dass Open–Source-Software im Bereich der Archivierung sinnvoll eingesetzt werden kann – einige Beispiele werden vorgestellt –, dass quelloffene Programme allein aber noch nicht die Probleme lösen können, die besonders mit der urheberrechtlichen Situation von Wissensmedien verbunden sind.

Eine ganz konkrete Art der Wissensverbreitung stellt Hans-Peter Merkel mit seinem Projekt *Linux4Afrika* vor. Bei diesem Projekt handelt es sich nur vordergründig um die Weitergabe gebrauchter und aufgearbeiteter Hardware an Partner des Projekts, Schulen, in Ostafrika. Das eigentliche Ziel der aufwändig von den freiwilligen Helfern implementierten Lösungen ist auch in diesem Fall die Schaffung von Möglichkeiten zur Partizipation am globalen Netz. So können kulturelle Schranken abgebaut und vielen jungen Menschen die Möglichkeit zum Wissenserwerb durch Teilnahme am Netz gegeben werden.

Dieses Kapitel und seine Artikel zeigen: Menschen, Wissen und Netze gehören zusammen. Schon Aristoteles stellte fest, dass alle Menschen von Natur aus nach Wissen streben. Die große Herausforderung unserer Zeit ist es, dieses individuell erworbene Wissen in verantwortlicher Weise in die von uns zum Transfer von Wissen und Informationen konstruierten Netze einzubringen, ohne das aufzugeben, was uns als privatem Raum wichtig und wertvoll ist. Wenn wir es schaffen, diese Balance auch in Zukunft zu halten, wird Alexander von Humboldt recht behalten:

> „Wissen und Erkennen sind die Freude und die Berechtigung der Menschheit; sie sind Teile des Nationalreichtums; oft ein Ersatz für die Güter, welche die Natur in allzu kärglichem Maße ausgeteilt hat."

Richard Stallmans Goldene Regel und das „Digital Commons“*

GLYN MOODY

Durch die allseits bekannten Erfolge freier Software beeinflussen jetzt auch allmählich die artverwandten Formen des Open Access, Open Data, Open Content etc. das öffentliche Bewusstsein. Dabei werden sie jedoch im Allgemeinen nur als bloße, wenn auch interessante, Anwendungen, d. h. als Imitationen, der Idee hinter freier Software gesehen. Dabei verliert man jedoch das große Ganze aus den Augen, denn durch die gemeinsamen Anstrengungen aller dieser Bewegungen entsteht ein riesiges, vollkommen neues *Digital Commons* des Wissens. Die größten Hindernisse bei der Ausweitung dieses *Commons* sind nunmehr eher rechtlicher als technischer Natur. Sie sind das Ergebnis politischer Lobbyarbeit durch die Content-Industrie, der es nicht gelungen ist, ihre Denkweise an eine digitale statt analoge Welt anzupassen. Die wachsende Wirtschaftlichkeit von Open-Source-Unternehmen, die ihre Software frei mit ihren Kunden teilen, zeigt neue Geschäftsmodelle auf, die das *Commons* annehmen, anstatt es einzuschränken.

Schlüsselwörter: Geistige Monopole · Digital Commons · Open Access · Open Data · Open Content · Teilen

1 Einleitung – Die goldene Regel

Als Richard Stallman 1983 sein GNU-Projekt[1] ankündigte, erklärte er seine Beweggründe dafür so:

> Ich denke, dass die „Goldene Regel“ es erfordert, dass ich, wenn ich ein Programm mag, es mit anderen Leuten, die es mögen, teilen muss.

* Aus dem Englischen von Janine Heinrich.

1 Siehe http://www.gnu.org/.

> Ich kann nicht guten Gewissens eine Nichtveröffentlichungserklärung oder eine Softwarelizenzerklärung unterschreiben. Damit ich weiterhin Computer benutzen kann, ohne meine Prinzipien zu verletzen, habe ich entschieden, einen ausreichenden Korpus an freier Software zusammenzustellen, sodass ich in der Lage sein werde, ohne jede nicht freie Software zurechtzukommen.[2]

1999 fügte er in einem Interview Folgendes hinzu:

> Das große Ziel [des GNU-Projekts] ist es, die Freiheit der Nutzer sicherzustellen, indem man ihnen freie Software zur Verfügung stellt und einen möglichst großen Spielraum für die Nutzung gänzlich freier Software bietet. Die Idee hinter GNU ist es ja gerade, den Menschen die Möglichkeit zu geben, mit ihren Computern zu arbeiten, ohne die Dominanz einer weiteren Instanz hinnehmen zu müssen. Sie sollen mit der Software arbeiten können, ohne, dass der Urheber der Software sagt: „Ich werde dir nicht zeigen, wie das funktioniert; ich werde dich vollständig von mir abhängig machen, und wenn du die Software mit deinen Freunden teilst, dann werde ich dich der Softwarepiraterie bezichtigen und ins Gefängnis stecken lassen." (Moody 2001)

Nach der Veröffentlichung des aufschlussreichen Essays „Die Kathedrale und der Basar" von Eric Raymond (1999), in dem gezeigt wird, warum freie Software so erfolgreich ist, hat man sich mehr auf das „Wie" von freier Software konzentriert. Man war so begeistert von der Feststellung, dass dieselben grundlegenden Methoden der dezentralisierten, gemeinschaftlichen Entwicklung auch auf Bereiche angewendet werden können, die weit über Software hinausgehen, sodass das „Warum" von freier Software – Stallmans „Goldene Regel" des Teilens – fast in Vergessenheit geriet.

Das ist zwar verständlich, aber trotzdem zu bedauern. Konzentriert man sich ausschließlich auf die Vielfalt der möglichen Anwendungsbereiche des Konzepts freier Software, so lässt man die Gemeinsamkeit außer Acht, die einem Großteil der Projekte zu Grunde liegt, nämlich zu teilen und einen uneingeschränkten Online-Zugang zu den größten Beständen des digitalen Wissens der Welt zu ermöglichen.

Die Rolle der freien Software bei diesem großen Vorhaben ist eindeutig. Da proprietäre Software eine Tatsache ist und normalerweise nicht geteilt oder untersucht werden kann, hat sich das GNU-Projekt die Aufgabe gestellt, diese Software nachzuahmen und für jedes ihrer Bestandteile freie und im Internet zugängliche Versionen zur Verfügung zu stellen, die der Benutzer anwenden, untersuchen und teilen kann. Andere Projekte, die nicht offiziell zum GNU-Projekt gehören – wie z. B. der Linux–Kernel – funktionieren auf dieselbe Weise. Fasst man diese Anstrengungen zusammen, so entsteht ein Korpus von freiem Code, der für jeden zugänglich ist. Es wird ein

2 Siehe http://groups.google.com/group/net.unix-wizards/msg/4dadd63a976019d7 [10. Feb. 2008].

Gemeingut gebildet, welches im Gegensatz zu den frühen Gemeingütern – normalerweise Ländereien zur freien Benutzung durch alle Anwohner – ein digitales und als solches gegen die „Tragedy of the Commons“ (Hardin 1968) resistent ist, da seine Ressourcen wiederholt und mit jedem geteilt werden können, ohne sie zu erschöpfen.

Die Erkenntnis, dass ein *Digital Commons* nach seiner Schaffung potenziell von jedem, der Zugriff darauf hat, genutzt und verwertet werden kann, fördert die gemeinsame Nutzung riesiger Bestände von sowohl alten als auch neuen Online-Inhalten (dies erfordert oft eine Digitalisierung). Zusammengefasst helfen sie bei der erstmaligen Entwicklung eines gigantischen *Digital Commons* des Wissens, basierend auf der Annahme, dass der Zugang dazu für jeden kostenlos und die Nutzung uneingeschränkt sein soll. Dies alles geschieht in dem Glauben, dass es letztlich zu einer gesellschaftlichen Bereicherung führt.

2 Aufbau von Digital Commons

Auch wenn Stallmans GNU-Projekt der bekannteste Versuch zur Entwicklung eines *Digital Commons* ist, war er bei Leibe nicht der erste. Diese Ehre gebührt Michael Hart und seinem *Project Gutenberg*[3]. Die Idee dazu wurde geboren, als Hart im Jahr 1971 Nutzungszeit für einen der ersten Großrechner erhielt. Er fragte sich, wie er dieses Geschenk am wirkungsvollsten nutzen könnte und erkannte, dass durch die Digitalisierung eines gedruckten Buchs jeder – oder zumindest die Nutzer des Netzwerks – ein Exemplar bekommen könnte, ohne dass Mehrkosten entstünden.

Aber wie Stallman war sich auch Hart bewusst, dass die vielfache Stärke, die man durch das Teilen digitaler Medien erlangt, so etwas wie eine moralische Verpflichtung begründete, da man mit so geringen Mitteln so viel erreichen kann. Die Aufgabe, so viele Bücher wie möglich zu digitalisieren und so weit wie möglich zu verteilen, nannte Hart *Project Gutenberg*, in der Hoffnung, dass sein Einfluss auf die Verbreitung und Demokratisierung des Wissens so bedeutend sein würde wie Gutenbergs Entwicklung des Buchdrucks.

Auch arbeitete Hart, wie Stallman, zu Beginn seines großen Projekts allein. Nach und nach schlossen sich ihm jedoch Gleichgesinnte, die sich von der Logik seines Vorhabens inspiriert sahen, an. Mit der Gründung der *Distributed Proofreaders*[4] im Jahr 2000 wurden einige Grundsätze der freien Software, wie z. B. die dezentrale Entwicklung separater Einheiten im gesamten Internet, eingeführt. 2005 startete man das Projekt *LibriVox*[5], welches Texte des *Project Gutenberg* in frei verfügbare Hörbücher umwandelt. Dies verdeutlicht, wie ein *Digital Commons* zur Entstehung eines vollkommen neuen *Digital Commons* führen kann.

3 Siehe http://www.gutenberg.org/wiki/Main_Page [10. Feb. 2008]
4 Siehe http://www.pgdp.net/c/ [10. Feb. 2008].
5 Siehe http://librivox.org/ [10. Feb. 2008].

Für die freie Verbreitung von Texten konnte das *Project Gutenberg* jedoch nur jene nutzen, die lizenzfrei erhältlich sind, da Urheberrechtsgesetze weltweit vorgeben, wie viel Zeit vergehen muss, bevor ein neu erschienenes Buch frei verfügbar wird. Als das Urheberrecht zu Beginn des 18. Jahrhunderts in England geschaffen wurde, betrug dieser Zeitraum 28 Jahre nach Veröffentlichung eines Werks.[6] Diese Schutzfrist wurde jedoch im letzen Jahrhundert in verschiedenen Ländern auf 50 oder sogar 70 Jahre nach dem Tod des Verfassers erweitert. Das führt dazu, dass zeitgenössische Texte nicht nur nicht aufgenommen werden können, sondern dass ein Großteil der Schriften des 20. Jahrhunderts nicht verfügbar sind. Das stellt einen massiven Verlust dar.

Die rechtlichen Einschränkungen, denen das *Digital Commons* – in diesem Fall Online-Texte – durch die ständige Verschärfung von Urheberrechtsgesetzen ausgesetzt ist, haben den bekannten Rechtsanwalt und Professor für Rechtswissenschaften Lawrence Lessig, dazu veranlasst, deren Rechtmäßigkeit vor einem amerikanischen Gericht anzufechten. Zwar verlor er den Fall, aber er fand eine Lösung, wie man den Schaden mit einer von Richard Stallman inspirierten Methode begrenzen könnte. Letzterer hatte die so genannte Copyleft-Lizenz (*GPL*[7]) erfunden und nutzte somit das Urheberrechtsgesetz, um sicherzustellen, dass sich niemand als Urheber der Codes des GNU-Projekts bezeichnen und sie in unfreie Codes umwandeln kann. Lessig gründete mit einigen anderen das *Copyright's Commons*, welches später in *Creative Commons*[8] (CC) umbenannt wurde und die Anwendung von neuen Creative–Commons-Lizenzen[9] überwacht.

Wie schon durch die GPL-Lizenzen, erhalten die Nutzer auch durch die CC–Lizenzen gewisse Sonderrechte, wie z. B. das Recht, Kopien anzufertigen oder das Recht, ein Werk frei zu verändern, um so die Einschränkungen durch die jüngsten Gesetzesänderungen zum Urheberrecht zu beseitigen. Somit haben sie die Möglichkeit, gemeinsam an Projekten zu arbeiten, so wie sie auch gemeinsam freie Software entwickeln können.

Auch Wissenschaftler bauen bekanntermaßen auf den Entdeckungen anderer, die normalerweise in Form von Aufsätzen in Zeitschriften veröffentlicht werden, auf.[10] Letztere sind unverzichtbar für diejenigen, die mit den neuesten Entwicklungen auf einem Gebiet Schritt halten wollen, können jedoch sehr teuer sein, da Abonnements für gewöhnlich tausende Euro pro Jahr kosten. Dadurch sind sie nicht nur für Forscher in Entwicklungsländern unerschwinglich, sondern auch für Universitätsbibliotheken im Westen, die mit einem knappen Budget auskommen müssen. Außerdem können nicht einmal jene Menschen, die einen Großteil der Forschung öffentlicher Einrichtungen finanzieren, nämlich die Steuerzahler, die Ergebnisse dieser Arbeit sehen.

6 Siehe http://www.copyrighthistory.com/anne.html.
7 Siehe http://www.gnu.org/copyleft/gpl.html [10. Feb. 2008].
8 Siehe http://creativecommons.org/ [10. Feb. 2008].
9 Siehe http://creativecommons.org/about/license/ [10. Feb. 2008].
10 Daher hat *Creative Commons* ein *Science Commons* gegründet (siehe http://sciencecommons.org/) [10. Feb. 2008].

Die Open-Access-Bewegung versucht genau dies zu ändern, indem sie ein *Digital Commons* für Forschungsergebnisse aufbaut, ohne jedoch wichtige Bestandteile traditioneller wissenschaftlicher Veröffentlichungen, wie z. B. die Begutachtung durch Fachkollegen, über Bord zu werfen.[11] Der Kernpunkt des Open-Access-Ansatzes ist die Veröffentlichung der Aufsätze im Internet, da die Kosten dafür verschwindend gering sind, während gedruckte Zeitschriften zwangsweise teurer herzustellen sind. Dadurch sind sie nicht mehr für die breite Masse zugänglich.

Der erste Versuch, wissenschaftliche Aufsätze in großem Rahmen elektronisch zu verbreiten, war das Projekt *arXiv.org*[12], bei dem Physiker ihre Vorabdrucke kostenlos ins Internet stellen können. Interessant hierbei ist, dass der Gründer dieses Projekts, Paul Ginsparg, schon 1985 mit dem GNU-Manifest vertraut war und bereits in den 1970er Jahren durch seinen Bruder, der damals am *MIT* studierte, von Stallman gehört hatte. Eine der wichtigsten Inspirationen für die Gründung der *Public Library of Science*[13] (*PLoS*) im Jahr 2001, welche einige der wichtigsten und bekanntesten Open–Access-Zeitschriften veröffentlicht, war Open-Source-Software. Die anderen beiden waren das Forum von *arXiv.org* und die öffentlich zugängliche Genomdatenbank.[14]

In letzterer werden die Ergebnisse jahrzehntelanger DNA-Sequenzierungen inklusive jener des Humangenomprojekts gespeichert. In der Bermuda-Konvention[15] von 1996 versprachen die Leiter der wichtigsten Forschungseinrichtungen auf dem Gebiet der Genomforschung, die vollständigen Ergebnisse ihrer Sequenzierungen immer sofort zu veröffentlichen, was bis dahin alles andere als üblich war. Hierbei ging es zum Teil darum, Unternehmen zuvorzukommen, die möglicherweise Patente auf Teile dieser Sequenzen anmelden könnten, indem sie etwas schaffen, das im Patentrecht als „Stand der Technik“ bezeichnet wird. Somit entstand ein *Genomic Commons*, welches ebenfalls digital ist, obgleich sein Code aus vier DNA-Basen (A, C, G, T) anstatt zwei Dualzahlen (0, 1) besteht.

Die öffentlichen Genomdatenbanken verdeutlichen sehr eindrucksvoll, wie man große Mengen an Rohdaten für die Öffentlichkeit zur Verfügung stellen und nutzbar machen kann. Dieses Teilen von Informationen ermöglicht immer ausgefeiltere Vergleiche von Gensequenzen, z. B. zwischen verschiedenen Arten, um Stammbäume zur Genomevolution zu erstellen oder zwischen verschiedenen Mitgliedern der selben Art, um kleinste Unterschiede aufzuzeigen, die im Zusammenhang mit Genkrankheiten stehen. Hierbei sind die Genomdatenbanken wichtige Wegbereiter für die Open-Data-Bewegung. Ihr Ziel ist es, alle Arten von digitalen Rohmaterialien für Verweise und *mashups* zu teilen. Besonders interessant sind in diesem Zusammenhang einerseits grundlegende wissenschaftliche Bestände, z. B. aus Physik, Chemie und Bio-

11 Die beste Einführung in dieses Gebiet findet man unter http://www.earlham.edu/~peters/fos/overview.htm [10. Feb. 2008].

12 Siehe http://arxiv.org/ [10. Feb. 2008].

13 Siehe http://www.plos.org/ [10. Feb. 2008].

14 Siehe z. B. http://www.ebi.ac.uk/embl/index.html [10. Feb. 2008].

15 Diskussion dazu unter http://www.sanger.ac.uk/HGP/policy-forum.shtml [10. Feb. 2008].

logie, und andererseits Datenbanken mit Informationen öffentlicher Einrichtungen, die keinem identifizierbaren Individuum zugeschrieben werden können.

In Großbritannien beispielsweise entwickelt eine staatliche Institution namens *Ordnance Survey* geographische Informationen, die sie verkauft, um ihre Aktivitäten zu finanzieren. Mit ihrer Kampagne *Free Our Data*[16] will die britische Zeitung *The Guardian* die Regierung dazu bewegen, diese Daten kostenlos verfügbar zu machen. Gleichzeitig gibt es Basisprojekte wie *OpenStreetMap*[17], welche in wahrer Open–Source-Tradition in Zusammenarbeit mit den Nutzern ihre eigenen Datensätze entwickeln wollen, genau wie es Stallman mit *GNU* tat. In diesem Fall werden die Daten mit preisgünstigen GPS-Geräten gesammelt und später kostenlos zur Verfügung gestellt. An einer anderen Stelle setzt das *Freebase-Projekt*[18], das sich selbst als „offene Datenbank der weltweiten Informationen" beschreibt, an. Es fordert die Menschen auf, Daten jeglicher Art ins Netz hochzuladen und beliebige Bemerkungen dazu abzugeben. Danach kann jeder komplexe Anfragen an die gesamte Datenbank stellen und mit Hilfe der Bemerkungen verschiedene Dateien verbinden oder große *mashups* erstellen.

Jedoch gibt es auch hier überraschenderweise rechtliche Probleme – zumindest in Europa. Die EU-Richtlinie über den rechtlichen Schutz von Datenbanken aus dem Jahr 1996 schaffte ein neues Recht für die Urheber von Datenbanken, die nicht unter das Urheberrecht fallen.[19] Sie sollte Investitionen in neue europäische Datenbanken fördern, da man davon ausging, dass ohne einen Schutz nur wenige Unternehmen dazu bereit wären.

In den 10 Jahren seit der Einführung der Richtlinie ist die Menge der Datenbanken in Europa dennoch nicht gestiegen. Vielmehr fiel deren Zahl nach Inkrafttreten der Richtlinie, während sie in den USA, wo es keinen derartigen Schutz für Datenbanken gibt, in die Höhe geschossen ist.[20] Dies zeigt, dass geistige Monopole, wie z. B. Urheberrechte, Patente, Datenbankrechte usw., kaum Anreize für innovatives Handeln darstellen.[21]

Ein wichtiger Grund für den Erfolg von Open Data, Open Source und Open Access ist der Netzwerkeffekt. Da die Anzahl der Personen, die zu diesen offenen Projekten beitragen, sehr hoch ist, vergrößert sich das (mathematische) Netz aus möglichen Interaktionen schneller als die Zahl der Beiträge. Das ist natürlich nur möglich, wenn man über ein Kommunikationsmittel wie das Internet verfügt, welches

16 Siehe http://www.freeourdata.org.uk/index.php [10. Feb. 2008].
17 Siehe http://www.openstreetmap.org/ [10. Feb. 2008].
18 Siehe http://www.freebase.com/ [10. Feb. 2008].
19 Siehe http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:31996L0009:DE:HTML (deutsche Version) [10. Feb. 2008].
20 Erster Evaluierungsbericht der Richtlinie 96/9/EC über den rechtlichen Schutz von Datenbanken verfügbar unter http://ec.europa.eu/internal_market/copyright/docs/databases/evaluation_report_en.pdf (englische Version) [10. Feb. 2008].
21 Siehe z. B. http://www.againstmonopoly.org/ [10. Feb. 2008].

einen problemlosen Kontakt zwischen den Teilnehmern und deren digitalen Inhalten ermöglicht.

3 Das Bedürfnis zu teilen und die Konflikte mit der Industrie

Die engen wechselseitigen Beziehungen zwischen geteilten Inhalten, Internet-Konnektivität und Kreativität werden ganz besonders durch die außergewöhnlich stark steigende Zahl der so genannten Web-2.0-Websites verdeutlicht. Einige, wie *Slashdot*, *Groklaw* und das Bloggen im Allgemeinen basieren auf der geteilten kollektiven Intelligenz ihrer Nutzer. Andere Seiten und deren Lizenzen, wie *Flickr*, *YouTube* und *Scribd* wurden speziell für das Teilen verschiedener Arten von offenen Inhalten entwickelt, wohingegen Seiten wie *MySpace*, *Facebook*, *LinkedIn* Interessen und Beziehungen ihrer Nutzer online aufzeigen und teilen. Wieder andere wie *Wikipedia*, *wikiHow* und *Wikitravel* gehen noch weiter und fordern die Nutzer auf, direkt zusammenzuarbeiten und geteiltes Material als Anfangspunkt zu nutzen. Alle diese Seiten gibt es erst, seit die Zahl der regelmäßigen Internetnutzer ein bestimmtes Niveau erreicht hat und die Geschwindigkeit der Verbindungen hoch genug ist, um vergleichsweise große Dateien, wie z. B. Videos, problemlos um die Welt zu schicken.

Allen diesen Seiten ist der Wunsch gemein etwas, zu teilen, seien es Informationen, Inhalte, persönliche Verbindungen oder in zunehmendem Maße eine Mischung aus allen dreien. Doch wie bereits das *Project Gutenberg* von unverhältnismäßigen Urheberrechtsbestimmungen behindert und freie Software von Patenten bedroht wird[22], so wird auch die Entwicklung dieser neuen Formen des *Digital Commons* von den rechtlichen Einschränkungen des amerikanischen *Digital Millennium Copyright Act* (*DMCA*) und der europäischen Urheberrechtsrichtlinie[23] ausgebremst. Beide bewirken eine starke Einschränkung der althergebrachten Fair-Use- beziehungsweise Fair-Dealing-Rechte.[24]

Die Spannungen zwischen dem, was Millionen von Nutzern täglich im Internet tun, und dem was die Gesetze ihnen erlauben, beruht auf einem fast angeborenen Verlangen zu teilen, ungeachtet dessen, ob es sich dabei um urheberrechtlich geschütztes Material – besonders bei Musik und Videos – handelt. Dieses Teilen geschieht selten zu kommerziellen Zwecken, denn in der Regel verkaufen die Nutzer die Inhalte nicht, sondern teilen sie einfach, weil sie denken, andere könnten sich dafür interessieren. Nach ihren Beweggründen gefragt, würden sie ihre Handlungen wahrscheinlich mit Stallmans „Goldener Regel“ rechtfertigen.

Oftmals sind die Handlungen derer, die dieser Regel folgen, sogar ein positives Marketing, durch das die Inhalte von Unternehmen unterstützt werden, und führen

22 Siehe Stallmans Diskussion unter http://www.cl.cam.ac.uk/~mgk25/stallman-patents.html [10. Feb. 2008].

23 Richtlinie 2001/29/EG zur Harmonisierung bestimmter Aspekte des Urheberrechts und der verwandten Schutzrechte in der Informationsgesellschaft.

24 Siehe z. B. Lessig (2002, S. 105).

zu höheren Verkaufszahlen. Dennoch investieren Medienkonzerne viel Zeit und Geld in die Bekämpfung dieser Art von Reklame, weil sie das Teilen ihrer Inhalte als direkten Umsatzverlust wahrnehmen und nicht als eine indirekte Methode für die deutliche Umsatzsteigerung.

Das Problem besteht zum Teil darin, dass Unternehmen, die digitale Güter anbieten, anders arbeiten müssen als jene, die analoge anbieten. Medienkonzerne haben bisher eher letztere angeboten, zuerst als Schallplatten und Kassetten und dann als CDs und DVDs. Während erstere noch eindeutig analog waren, so sind CDs und DVDs eine Mischung aus beiden, d. h. der Inhalt ist digital, aber das Format analog. Medienkonzerne haben sich verständlicherweise mehr auf den Inhalt konzentriert, was jedoch zu einer Unterbewertung der analogen Aspekte ihrer Waren führte.

Versuche, Inhalte mit *DRM* (*Digital Rights Management* oder wie Stallman sagt *Digital Restrictions Management*[25]) zu schützen, sind im Internetzeitalter vergeblich, da man den Kopierschutz nur ein einziges Mal irgendwo auf der Welt durchbrechen muss, um innerhalb weniger Stunden DRM-freie Kopien im Internet zu erhalten. Da die Herstellungskosten für digitale Inhalte praktisch null sind, kann man nur schwer einen deutlich höheren Preis aufrechterhalten, besonders wenn es zusätzliche Unannehmlichkeiten wie DRM gibt. Auch wenn dieses Problem für gewinnorientierte Unternehmen aus der Content-Industrie unüberwindbar erscheint, so hat die Open-Source-Welt doch gezeigt, wie man das Teilen digitaler Inhalte erlauben, sogar unterstützen und trotzdem Profit machen kann.

Die ersten Unternehmen der Open-Source-Branche wie *Red Hat* entwickelten ein Abonnementsystem[26], bei dem zwar der Code öffentlich gemacht wird, aber der Kundendienst und andere Hilfsleistungen kostenpflichtig sind. Eine neue Generation von Unternehmen, die freie Software für Unternehmensanwendungen anbieten, wie z. B. *MySQL* (Datenbanken), *Alfresco* (unternehmensorientiertes Content-Management), *JasperSoft* (Business Intelligence) etc., beweisen, dass das Konzept von *Red Hat* auch weit über den Bereich der Betriebssysteme hinaus anwendbar ist.

Auf den ersten Blick scheint nicht klar, wie man diesen Ansatz auch auf andere Bereiche, z. B. Musik, anwenden kann. Der springende Punkt ist jedoch, dass Open–Source-Unternehmen ihr Geld mit knappen Gütern, wie Kundensupport, Beratung etc., verdienen und nicht mit jenen, die von jedem problemlos tausendfach kopiert werden können (dem Code). Das Geheimnis beim Finden brauchbarer Geschäftsmodelle liegt darin, sich darauf zu konzentrieren, was man dem Kunden bieten kann, das sich dieser nicht woanders im Internet beschaffen könnte.

Vor kurzem begannen einige bekannte Musiker, Methoden zu ergründen, mit denen sie ihre Musik jedem, der sie haben möchte, zur Verfügung stellen können und dabei trotzdem Geld verdienen. Der vielleicht beste Ansatz ist jener der Band *Radiohead*, die es den Nutzern gestattet, selbst zu entscheiden, wieviel sie für den Download ihres

25 Siehe http://www.defectivebydesign.org/what_is_drm [10. Feb. 2008].
26 Siehe http://www.redhat.com/software/subscriptions.html [10. Feb. 2008].

neuesten Albums *In Rainbows* bezahlen wollen, was auch bedeutet, dass sie nichts zahlen müssen.

Radiohead fügten diesem neuartigen Preissystem eine weitere Einkommensmöglichkeit hinzu, die sich aus dem Verkauf rein analoger Produkte ergibt. Sie bieten etwas an, das sie *Discbox*[27] nennen. Diese beinhaltet eine CD-Version ihrer Musik, zwei Schallplatten, eine erweiterte CD mit Bonustracks, Kunst, Photos und Liedtexten, wird in einem gebundenen Buch und einer Schutzhülle geliefert und kostet 40 Pfund. Da analoge Güter (noch) nicht einwandfrei kopiert werden können, werden die Fans ganz automatisch diese Produkte mit Bonusmaterial kaufen, auch oder vielleicht gerade dann, wenn sie sich entschließen, die Musik kostenlos herunterzuladen.

Auch erkennen Musiker mittlerweile, dass sie oftmals mehr Geld mit Konzerttouren verdienen können als mit dem Verkauf von CDs, und viele tun dies bereits.[28] Die Gründe dafür sind schnell gefunden: Konzerte sind hochemotionale Ereignisse und soziale Erfahrungen, die nicht durch das bloße, isolierte Hören derselben Musik erzeugt werden können. Deshalb kann man so hohe Preise für Eintrittskarten verlangen. Der nächste logische Schritt wäre es, CDs als Werbeprodukte für teure Konzerttouren und Merchandising zu verschenken, was Künstler wie u. a. *Prince* bereits tun.[29]

Dies setzt ein Zeichen für die Filmindustrie, die beispielsweise mit der kollektiven Erfahrung, Filme im Kino zu sehen, Geld verdienen kann. Diese analoge und hochgradig persönliche Erfahrung unterscheidet sich grundlegend vom Anschauen einer DVD oder heruntergeladenen Filmen. Mit dem richtigen Konzept kann man in einem solchen Superkino deutlich höhere Preise verlangen und die digitalen Versionen der Filme als kostenlose Lockmittel für das „Original“ nutzen.

Auch im Verlagswesen, das von so manchen in einer Welt der digitalen Inhalte bereits als hoffnungslos veraltet angesehen wird, lässt sich diese Idee anwenden. Die Texte der Bücher könnten kostengünstig im Internet angeboten werden, um die Leser dazu zu bewegen, Lesungen, bei denen die Autoren auch Fragen beantworten und gedruckte Versionen signieren, zu besuchen – was auch sehr persönliche und analoge Erfahrungen sind. Außerdem könnte man stark auf den Kunden zugeschnittene Auflagen herausbringen, bei denen der Schwerpunkt auf physischer Qualität, wie z. B. Papierqualität, hochwertigen Einbänden etc., liegt. Der Vatikan als Verleger hat dieses Prinzip scheinbar verstanden: Er veröffentlichte kürzlich eine historische Handschrift aus seinen Gewölben, die auf reproduziertem Pergament gedruckt, mit einem imitierten Wachssiegel des Papstes versehen und zusammen mit einem wissenschaftlichen Kommentar in einem weichen Lederetui verpackt ist.[30] Der eigentliche Inhalt der Handschrift verblasst neben diesen Zugaben. Der Preis? Nur etwas mehr als 5 900 Euro.

27 Siehe http://www.waste.uk.com/Store/waste-radiohead-dii-11-10023-discbox+audio.html [10. Feb. 2008].

28 Siehe http://news.bbc.co.uk/1/hi/business/4896262.stm [10. Feb. 2008].

29 Siehe http://www.economist.com/business/displaystory.cfm?story_id=9443082 [10. Feb. 2008].

30 Siehe http://news.bbc.co.uk/1/hi/world/europe/7044741.stm [10. Feb. 2008].

4 Fazit

Alle diese wirtschaftlichen Betrachtungen scheinen nichts mehr mit Richard Stallmans „Goldener Regel" und seinem Wunsch, Nutzerfreiheit durch Software zu ermöglichen, zu tun zu haben. Heute sind jedoch die Haupthindernisse bei der Ausschöpfung des vollen Potenzials von Inhalten keine technischen, wie zu den Anfängen von Harts oder Stallmans wegweisenden Projekten, sondern größtenteils rechtliche, begründet in umfangreicher politischer Lobbyarbeit von Unternehmen, die Angst vor der Zukunft haben und an altmodischen Geschäftsmodellen festhalten.

Bis diese Industriezweige die Möglichkeiten der digitalen Welt schätzen lernen und die Gesetzgebung weltweit den Realitäten der Internetgesellschaft angepasst wird, werden große Bereiche des Wissens, welche ein potenzieller Teil des großen bereits heute von Millionen von Freiwilligen geschaffenen *Digital Commons* sind, weiterhin von der „Tragedy of the Anti-Commons" (riesige, verschwendete Bereiche, in denen Stallmans „Goldene Regel" nicht angewendet wird) befallen sein (Heller und Eisenberg 1998).

Literatur

Hardin, G. (1968), 'The Tragedy of the Commons', *Science Magazine* **162**(3859), S. 1243–1248.

Heller, M. A. und Eisenberg, R. S. (1998), 'Can Patents Deter Innovation? The Anticommons in Biomedical Research', *Science Magazine* **280**(5364), S. 698–701.

Lessig, L. (2002), *The Future of Ideas*, Vintage Books, New York.

Moody, G. (2001), *Rebel Code: Linux and the Open Source Revolution*, Penguin Books, London.

Raymond, E. S. (1999), *The Cathedral & the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*, O'Reilly, Sebastopol, California.

Archivierung und Open Source

CHRISTOPH JEGGLE UND ULRICH KAMPFFMEYER

Nach einem Überblick über die Bedeutung der Archivierung und dabei insbesondere der elektronischen Archivierung wird der Beitrag von Open Source und Open Content zur Archivierung beleuchtet. Dabei geht es um die unterschiedlichen Arten von Open-Source-Software im Bereich der Archivierung, aber auch um die Chancen von Open Source und Open Content für die Lösung urheberrechtlicher Herausforderungen. Die Open-Source-Community muss sich aber auch der kulturellen Verantwortung für die Archivierung digitaler Information stellen und ihren Beitrag dazu leisten.

Schlüsselwörter: Harvester-Software · Elektronisches Archiv · Archivierung · Records Management

1 Archivierung – eine Begriffserklärung

Der Begriff Archivierung wird sehr unterschiedlich benutzt. Er ist abgeleitet vom Begriff (lat.) *archivum* aus (griech.) *archeío(n)*, „Regierungs-" oder „Amtsgebäude". Er bezeichnet sowohl die Institution wie auch den Gegenstand des Archivs, die geordnete Sammlung von Aufzeichnungen. Archive dienen zur langzeitigen und sicheren Bewahrung von aufbewahrungswürdigen oder aufbewahrungspflichtigen Objekten. In der physischen Welt können dies Dokumente, Akten und Bücher, aber auch andere Medien und Materialen sein. Die gespeicherten Objekte werden über Registraturen, Findbücher, Verzeichnisse oder moderne Datenbanken verwaltet und erschlossen. Unter Langzeitarchivierung versteht der klassische Archivar die Bewahrung für die Ewigkeit.

1.1 Elektronische Archivierung

In der elektronischen Welt wird der Begriff Archivierung je nach Sprachgebrauch für Datensicherung oder datenbankgestützte Archivierung verwendet. Hier machen sich

auch Unterschiede in der Sprache bemerkbar, wenn es um die Übertragung von englischen Begriffen für Software und Speichertechnologien geht: *Archiving* steht eher für Datensicherung und Backup, wohingegen *Digital Preservation* unserem Verständnis von Langzeitarchivierung entspricht. Der Begriff *elektronische Archivierung* steht für die unveränderbare, langzeitige Aufbewahrung elektronischer Information und wird benutzt, um die datenbankgestützte unveränderbare Speicherung von Informationen zu beschreiben (Kampffmeyer und Rogalla 1997). Anders als bei der Datensicherung geht es hierbei darum, den Zugriff auf Einzelobjekte über Suchanfragen zu ermöglichen und die gefundene Information unabhängig vom ursprünglichen System wieder verfügbar zu machen. Die Verwaltung der Objekte wird dabei in der Regel getrennt von ihrer Speicherung auf elektronischen Medien betrachtet. Zur Langzeitaufbewahrung gehören die regelmäßige kontinuierliche Prüfung der Lesbarkeit und Verfügbarkeit der Information sowie die rechtzeitige oder kontinuierliche, verlustfreie Migration der gespeicherten Informationsobjekte.

Elektronische Archivsysteme sind durch folgende Eigenschaften gekennzeichnet (Kampffmeyer 2004):

1. Programmgestützter, direkter Zugriff auf einzelne Informationsobjekte, landläufig auch Dokumente genannt, oder Informationskollektionen, z. B. Listen, Container mit mehreren Objekten etc.,

2. datenbankgestützte Verwaltung der Informationsobjekte auf Basis von Metadaten und gegebenenfalls Volltexterschließung der Inhalte der archivierten Informationsobjekte,

3. Unterstützung verschiedener Indizierungs- und Recherchestrategien, um auf die gesuchte Information direkt zugreifen zu können,

4. einheitliche und gemeinsame Speicherung beliebiger Informationsobjekte, vom gescannten Faksimile über Office-Dateien und E-Mails bis hin zu komplexen XML-Strukturen, Listen, COLD-Dokumenten (*Computer Output on Laserdisk*) oder ganzen Datenbankinhalten,

5. Verwaltung von Speichersystemen mit nur einmal beschreibbaren Medien einschließlich des Zugriffs auf Medien, die sich nicht mehr direkt im Speichersystem befinden,

6. Sicherstellung der Verfügbarkeit der gespeicherten Informationen über einen längeren Zeitraum, der Jahrzehnte betragen kann,

7. Bereitstellung von Informationsobjekten unabhängig von der sie ursprünglich erzeugenden Anwendung auf verschiedenen Klienten und mit Übergabe an andere Programme,

8. Unterstützung von *Klassenkonzepten* zur Vereinfachung der Erfassung durch Vererbung von Merkmalen und Strukturierung der Informationsbasis,

9. Konverter zur Erzeugung von langfristig stabilen Archivformaten und *viewer* zur Anzeige von Informationsobjekten, für die die ursprünglich erzeugende Anwendung nicht mehr zur Verfügung steht,

10. Absicherung der gespeicherten Informationsobjekte gegen unberechtigten Zugriff und gegen Veränderbarkeit der gespeicherten Information,

11. übergreifende Verwaltung unterschiedlicher Speichersysteme, um z. B. durch Zwischenspeicher (*caches*) schnellen Zugriff und zügige Bereitstellung der Informationen zu gewährleisten,

12. standardisierte Schnittstellen, um elektronische Archive als Dienste in beliebige Anwendungen integrieren zu können,

13. eigenständige Wiederherstellungsfunktionalität (*recovery*), um inkonsistent gewordene oder gestörte Systeme aus sich heraus verlustfrei wiederaufbauen zu können,

14. sichere Protokollierung von allen Veränderungen an Strukturen und Informationsobjekten, die die Konsistenz und Wiederauffindbarkeit gefährden können und dokumentieren, wie die Informationen im Archivsystem verarbeitet wurden,

15. Unterstützung von Standards für die spezielle Aufzeichnung von Informationen auf Speichern mit WORM-Verfahren (*write once read multiple*), für gespeicherte Dokumente und für die Informationsobjekte beschreibende Metadaten, um eine langfristige Verfügbarkeit und die Migrationssicherheit zu gewährleisten,

16. Unterstützung von automatisierten, nachvollziehbaren und verlustfreien Migrationsverfahren.

1.2 Unterschiedliche Anwendungsgebiete

Bei der elektronischen Archivierung sind folgende Anwendungsgebiete zu unterscheiden, die entsprechende Produkteigenschaften und Implementationen voraussetzen.

Aufbewahrungspflichtige Informationsobjekte

Archive werden als reine Dokumentations- und Sicherungssysteme zur Erfüllung rechtlicher und regulativer Anforderungen eingesetzt. Der Maßstab für die Dauer und die Form der Archivierung wird durch Gesetze und Verordnungen vorgegeben.

Archivierung geschäftlicher Unterlagen

Hier wird in der Regel von der *revisionssicheren Archivierung* gesprochen (Kampffmeyer und Rogalla 1997). Für die Revisionssicherheit gelten die Kriterien Ordnungsmäßigkeit, Vollständigkeit, Sicherheit des Gesamtverfahrens, Schutz vor Veränderung und Verfälschung, Sicherung vor Verlust, Nutzung nur durch Berechtigte, Einhaltung der Aufbewahrungsfristen, Dokumentation des Verfahrens, Nachvollziehbarkeit und Prüfbarkeit. Diese aus dem Handelsgesetz und den steuerlichen Gesetzen abgeleiteten Anforderungen wurden inzwischen auch auf andere Anwendungsgebiete übertragen. Die Archivierungszeiträume betragen hier in der Regel sechs bis elf Jahre, können aber auch erheblich länger werden.

Archivierung von Kunden-, Produkt- und technischen Unterlagen

Hier sind in der Regel zahlreiche branchenspezifische Anforderungen zur *compliance* zu berücksichtigen, die Zeiträume von mehreren Jahrzehnten und mehr abdecken müssen. Anforderungen von Krankenhäusern, Kraftwerken und anderen Organisationen gehen über 100 Jahre hinaus. Für sie gelten auch die Anforderungen der revisionssicheren Archivierung.

Aufbewahrungswürdige Informationsobjekte

Hierunter fallen Archive, die aktiv genutzt werden und bei denen der Dokumentationsaspekt zu vernachlässigen ist. Die Dauer und die Form der Archivierung werden durch das Unternehmen oder die Organisation selbst bestimmt.

Archivierung für das Wissensmanagement

Große Mengen von Informationen sollen für die Erschließung und aktive Nutzung aufbewahrt werden. Neben die traditionelle Erschließung mit feldorientierter Suche über Metadaten treten hier Volltexterschließung, Suchmaschinen, *Tagging*, semantische Netze und andere Strategien. Revisionssichere Archivmedien spielen nur eine nachgeordnete Rolle zum „Einfrieren" von Informationen und temporären Informationszusammenhängen. Auch *Data-Warehouses* oder *Information-Warehouses* können dieser Kategorie zugerechnet werden.

Archivierung im E-Business

Informationen werden in großen Archiven für die Nutzung durch Dritte bereitgehalten. Hierzu zählen auch Webseiten, die Musik, Filme, Dokumente oder andere Informationen gegen Bezahlung anbieten. Archivsysteme bieten hier die Möglichkeit, den so genannten *Longtail* zu bedienen und auch selten nachgefragte Informationen bereitzuhalten. In solchen Systemen spielen neben der sicheren Speicherung auch Aspekte des *Digital Rights Management* eine wichtige Rolle.

Archivierung als Service für jedermann

Zunehmend werden Archivierungsangebote als *Software as a Service* (SaaS) oder *Application Service Provider* (ASP) über das Internet verfügbar gemacht. Sie zielen vorrangig auf Privatleute und kleinere Unternehmen, die sich keine aufwändigen hausinternen Archivsysteme leisten können. Daneben gibt es aber auch ASP-Archivsystemangebote für Unternehmen, die die Anforderungen der Revisionssicherheit erfüllen.

Historische Archivierung und kulturelles Erbe

Auch wenn es in großen Wirtschaftsunternehmen Firmenarchive gibt, so ist doch das eigentliche Anwendungsgebiet der Langzeitarchivierung in den historischen Archiven, Bibliotheken und Museen zu sehen. Diese Organisationen legen gänzlich andere Maßstäbe an die Langzeitarchivierung an, als dies bei herkömmlichen elektronischen Archivsystemen der Fall ist. Die Dauer und die Form der Archivierung werden zum Teil durch Gesetze, besonders aber durch die Aufgabe der jeweiligen Institution bestimmt. Die Archive werden für die Nutzung durch offene Benutzergemeinschaften ausgelegt.

1.3 Records Management und elektronische Archivierung

Das Thema *Records Management* hat besondere Bedeutung für die Nutzung der Inhalte von elektronischen Archiven. Records Management dient zur inhaltlichen Erschließung der gespeicherten Informationen. Records-Management-Lösungen verwalten nicht nur elektronische Informationsobjekte, sondern auch Referenzen zu physischen Objekten und ihren Speicherlokationen. In der Norm ISO-15489 wird *Records Management* definiert als effiziente und systematische Kontrolle und Durchführung der Erstellung, Entgegennahme, Aufbewahrung, Nutzung und Aussonderung von Schriftgut einschließlich der Vorgänge zur Erfassung und Aufbewahrung von Nachweisen und Informationen über Geschäftsabläufe und Transaktionen in Form von Akten. Dies ist als Führungsaufgabe wahrzunehmen.

Dieser Bereich ist stark durch rechtliche Vorschriften und Standards geregelt. Daher wird in diesem Abschnitt ein kurzer Überblick über diese Vorschriften und Standards in Deutschland gegeben. Auf diese wird im Abschnitt 2 noch einmal eingegangen, um zu zeigen, welche Chancen Open Source auch für die Erfüllung rechtlicher Anforderungen bietet.

Neben der ISO-Norm gibt es zahlreiche andere Normen, Gesetze und Vorschriften, die das *Records Management* regeln. Kaum ein Anwendungsgebiet ist so durch nationale und internationale Normen, Gesetze und *Codes of Best Practice* standardisiert wie das *Records Management*. Mit den *Model Requirements for Electronic Document and Records Management* (MoReq und MoReq2) (Kampffmeyer 2007*b*) gibt es einen europäischen Standard für das *Records Management*. Zugrunde liegt ein Referenzmodell, das Eingangs-, Archiv- und Bereitstellungsobjekte unterscheidet.

Die Kombination eines Archivspeichers mit einem Archivsystem und einem Records–Management-System liefert erst zusammengenommen die notwendige Funktionalität für ein elektronisches Archivsystem. Nur so lassen sich die zahlreichen rechtlichen Anforderungen an die Speicherung von Informationen erfüllen. So fordern z. B. die Grundsätze zum Datenzugriff und zur Prüfbarkeit digitaler Unterlagen (GDPdU) in Deutschland, dass Informationen auswertbar aufbewahrt werden müssen und nicht in PDF oder Tiff konvertiert werden dürfen (Kampffmeyer 2005*a*). Die Grundsätze ordnungsmäßiger DV-gestützter Buchführungssysteme (GoBS) regeln das Erfassen von Datenströmen und das Scannen von Dokumenten. In den GoBS sind auch die Anforderungen an das interne Sicherheitssystem und die Verfahrensdokumentation festgelegt (Kampffmeyer und Rogalla 1997). Das Signaturgesetz beschreibt nicht nur den Einsatz von elektronischen Signaturen, sondern setzt zugleich implizit die Bedingungen für die langzeitige sichere Speicherung elektronisch signierter Dokumente. Das Bürgerliche Gesetzbuch (BGB) und die Zivilprozessordnung (ZPO) definieren den rechtlichen Charakter von elektronischen Dokumenten und damit auch letztlich die Bedingungen, wie diese Dokumente – einschließlich E-Mails und andere Formen von Dokumenten – sicher aufbewahrt werden müssen (Kampffmeyer 2005*b*). Durch die Angleichung der rechtlichen Vorgaben der Papierwelt an die elektronische Welt kommen immer mehr Vorschriften zum Tragen, die die elektronische Archivierung erforderlich machen. Zugleich erhöhen sich aber ständig die Anforderungen an die funktionalen Eigenschaften der Systeme, um diesen Anforderungen gerecht zu werden (Kampffmeyer 2007*a*).

Die grundsätzlichen Anforderungen an die elektronische Archivierung hat der VOI – Verband Organisations- und Informationssysteme e.V. wie folgt in zehn Merksätzen zusammengefasst (Kampffmeyer und Rogalla 1997):

1. Jedes Dokument muss unveränderbar archiviert werden.
2. Es darf kein Dokument auf dem Weg ins Archiv oder im Archiv selbst verloren gehen.
3. Jedes Dokument muss mit geeigneten *Retrieval-Techniken* wieder auffindbar sein.
4. Es muss genau das Dokument wiedergefunden werden, das gesucht worden ist.
5. Kein Dokument darf während seiner vorgesehenen Lebenszeit zerstört werden können.
6. Jedes Dokument muss in genau der gleichen Form, wie es erfasst wurde, wieder angezeigt und gedruckt werden können.
7. Jedes Dokument muss zeitnah wiedergefunden werden können.

8. Alle Aktionen im Archiv, die Veränderungen in der Organisation und Struktur bewirken, sind derart zu protokollieren, dass die Wiederherstellung des ursprünglichen Zustands möglich ist.

9. Elektronische Archive sind so auszulegen, dass eine Migration auf neue Plattformen, Medien, Softwareversionen und Komponenten ohne Informationsverlust möglich ist.

10. Das System muss dem Anwender die Möglichkeit bieten, die gesetzlichen Bestimmungen sowie die betrieblichen Bestimmungen des Anwenders hinsichtlich Datensicherheit und Datenschutz über die Lebensdauer des Archivs sicherzustellen.

Diesen Herausforderungen haben sich bisher nur wenige Open-Source-Produkte gestellt. Die wenigsten zielten dabei auf den Bereich der kommerziellen revisionssicheren Archivierung, sondern eher auf die Archivierung von Webinhalten und die Speicherung von Forschungsergebnissen und kulturellen Hinterlassenschaften. Damit lassen sich auch zugleich zwei unterschiedliche Treiber der Entwicklung feststellen – auf der einen Seite Universitäten, Museen, Archive und Bibliotheken, auf der anderen Seite Unternehmen mit kommerziellen Lösungen. Die Lösungen sehen dementsprechend sehr unterschiedlich aus.

2 Open Source – eine Chance für die Archivierung

Nachdem in den vorangegangenen Abschnitten die Bedeutung der Archivierung dargestellt wurde, kommen wir in diesem Abschnitt zu den Chancen, die das Open–Source-Prinzip für die Archivierung darstellt.

Dabei wird Open Source nicht nur im engeren Sinn als Prinzip der Softwareentwicklung und -lizenzierung verstanden, sondern im weiteren Sinne auch als *Open Content*. Mit Open Content werden nicht nur Quelltexte, sondern auch Inhalte allgemein lizenzgebührfrei zur allgemeinen Verwertung zur Verfügung gestellt. Open Content steht damit im Gegensatz zum Urheberrecht, durch das die Verwendung von Inhalten im Interesse des Urhebers eingeschränkt wird.

Archivierung im Sinne der Bewahrung der Inhalte zur allgemeinen Verwertung muss das Urheberrecht, das genau diese allgemeine Verwertung einschränkt, berücksichtigen.

2.1 Archivierung und Copyright

Archivierung bewegt sich nicht in einem rechtsfreien Raum. Das gilt sowohl für die Aufgabe der Archivierung selbst als auch für den Gegenstand der Archivierung. Beides hat Bedeutung für die Archivierung. Die rechtlichen Anforderungen, die für die Archivierung selbst gelten, sind bereits im Abschnitt 1.3 behandelt worden.

Die Archivierung von Inhalten, um diese langfristig einer Öffentlichkeit zugänglich zu machen, berührt den Schutz dieser Inhalte durch das Urheberrecht beziehungsweise Copyright. Projekte wie das *Internet Archive* und die *Open Content Alliance* müssen das berücksichtigen.

Das Internet Archive hat es sich zur Aufgabe gesetzt, Inhalte von Webseiten, die üblicherweise nur von sehr kurzer Lebensdauer sind, zu bewahren, gewissermaßen als elektronische Bibliothek des Weltwissens und der Weltkultur. Die meisten Inhalte, die dabei im Internet archiviert werden, unterliegen aber dem Urheberrecht beziehungsweise Copyright. Im Grunde können nur Inhalte archiviert werden, deren Schutz durch das Urheberrecht abgelaufen ist beziehungsweise für die der Urheber oder Inhaber der Rechte die Erlaubnis gegeben hat, sie in das Archiv aufzunehmen.

So archiviert die *Open Content Alliance* nur digitalisierten Text und Multimedia-Dokumente elektronisch, die keinem Copyright mehr unterliegen oder bei denen der Copyrightinhaber der Archivierung ausdrücklich zugestimmt hat. Dabei kann der Rechteinhaber auch Einschränkungen hinsichtlich der Nutzung des archivierten Materials durch die Anwender bestimmen.

Verschärft wird die Problematik von Archivierung und Copyright durch die Veränderung des Copyright im Jahr 1976. Bis dahin gab es in den USA ein *Opt-In Copyright*. Das heißt, dass das Copyright nicht automatisch vergeben wurde, sondern vom Urheber angemeldet werden musste. Diese Anmeldung musste nach einer gewissen Zeit wiederholt werden, um sicherzustellen, dass der Urheber weiterhin das Copyright beanspruchte. Tat er es nicht, wurde das Werk automatisch zur *Public Domain*. Dieses Opt-In Copyright wurde 1976 zu einem *Opt-Out Copyright*. Jedes Werk unterliegt automatisch dem Copyright. Eine Verlängerung des Copyright ist nicht erforderlich. Damit gibt es eine große Zahl von Werken, die kommerziell keine Bedeutung mehr haben, für die aktiv auch niemand mehr das Copyright beansprucht, die aber weiterhin noch dem Copyright unterliegen, weil niemand ausdrücklich auf das Copyright verzichtet hat. Eine Klage des Initiators des *Internet Archive*, Brewster Kahle, gegen das Opt-Out Copyright wurde vom Supreme Court abgewiesen.[1]

Inhalte, die nicht dem Copyright, sondern dem *Copyleft* unterliegen, sind als Open Content bewusst von den Urhebern für die allgemeine Nutzung und damit auch Archivierung freigegeben.

Insofern unterstützt Open Source im Sinne von Open Content die Bemühungen, das Wissen und die Kultur unserer Zeit nicht dem Vergessen zu überlassen, sondern mit allen technischen Möglichkeiten zu erfassen und dauerhaft aufzubewahren.[2]

1 Siehe http://cyberlaw.stanford.edu/node/5110 [4. Jan. 2008].

2 Zum Internet Archive und der Open Content Alliance siehe auch Heuer (2006), zu den rechtlichen Rahmenbedingungen siehe Spindler (2006*a*).

2.2 Open-Source-Anwendungen für die digitale Archivierung

Software

Trotz des oben erwähnten Akzeptanzproblems von Open-Source-Software für die Archivierung gibt es eine Reihe von solchen Produkten. In den meisten Fällen handelt es sich um Software aus dem universitären Umfeld, mit deren Hilfe Informationen aus Forschung und Kultur langfristig einem bestimmten Benutzerkreis oder sogar der allgemeinen Öffentlichkeit zugänglich gemacht werden sollen.

Diese Software deckt nur einen Teil der im Abschnitt 1 genannten Anforderungen an *Records Management* und elektronische Archivierung ab. Der Schwerpunkt dieser Anwendungen liegt in der Verwendung der Metadaten für eine möglichst einheitliche Strukturierung der archivierten Dokumente. Damit wird eine einfache Suche nach den Dokumenten im Archiv ermöglicht. Gleichzeitig ermöglicht die einheitliche Struktur eine gemeinsame Nutzung mehrerer Archive.

Der Aspekt der Langzeitaufbewahrung von Dokumenten und die dafür notwendige Unterstützung entsprechender Hardwareinfrastruktur wird bei Open-Source-Produkten aus dem universitären Umfeld weniger beachtet. Eine Langzeitaufbewahrung ist zwar möglich, aber diese Funktionalität steht nicht im Vordergrund.

Aus dem kommerziellen Umfeld kommen aber zunehmend Produkte, die eher den Anforderungen des *Records Management* gerecht werden und gleichzeitig auch als Dokumentenmanagementsystem verwendet werden können. Dadurch decken diese Open-Source-Produkte den gesamten Lebenszyklus von Dokumenten von der Erfassung bis zur Entsorgung nach Ablauf der Aufbewahrungszeiten ab.

Open-Source-Software bietet sich für die Abbildung des gesamten Lebenszyklus von Dokumenten an. Da dieser Lebenszyklus viele Jahre für ein Dokument betragen kann, birgt proprietäre Software das Risiko einer hohen Abhängigkeit von einem Hersteller. Migrationen von Daten, die bei einer Langzeitarchivierung unumgänglich sind, werden bei Open-Source-Software durch die vollständige Dokumentation einschließlich Quelltext erleichtert. Dieser Vorteil wird noch verstärkt, wenn sich die Open-Source-Software an offene Standards hält.

Selbstverständlich bedeutet auch ein Open-Source-System eine Abhängigkeit, aber die Möglichkeit, auf den Quelltext zuzugreifen und gegebenenfalls auch Änderungen und Anpassungen am Programm auch unabhängig vom „Hersteller" machen zu können, bedeutet noch eine signifikante Reduzierung dieser Abhängigkeit. Dieses Argument wird zwar bei jeder Art von Software für Open Source angeführt, aber es hat bei Software aus dem Bereich Dokumentenmanagement und Archivierung einen deutlich höheren Stellenwert, da es erheblich leichter ist, eine Office-Anwendung auszutauschen, als ein langjährig verwendetes *Repository* für Dokumente und Informationen zu wechseln.

Ein weiterer Aspekt ist die Notwendigkeit einer Verfahrensdokumentation, die für die Aufbewahrung von steuerlich relevanten Unterlagen unter anderem in Deutsch-

land vorgeschrieben ist. Ein Problem bei der Erstellung einer solchen Verfahrensdokumentation, mit der nachgewiesen werden muss, dass das Verfahren zur Erfassung, Bearbeitung und Aufbewahrung von steuerlich relevanten Dokumenten und Informationen revisionssicher, zuverlässig und gegen Manipulation geschützt ist, ist, dass man häufig auf die Unterstützung durch den Hersteller proprietärer Software angewiesen ist, um die inneren Mechanismen der verwendeten proprietären Software zu verstehen und dokumentieren zu können.

Bei Open-Source-Produkten ist solch eine Unterstützung durch den „Hersteller" sicherlich auch hilfreich und meist durch Foren der Open-Source-Community auch sichergestellt. Dennoch ist niemand auf eine solche Unterstützung angewiesen, da es sich um offene Software handelt, bei der im äußersten Fall im Source Code die interne Funktionsweise nachvollzogen werden kann.

Ein weiterer Aspekt bei der Archivierung digitaler Informationsobjekte ist das Format dieser Objekte. Viele Informationsobjekte liegen in einem proprietären Format vor. Ein Beispiel dafür sind die binären Dateiformate des Microsoft-Office-Paketes. Diese Informationsobjekte können ohne die zugehörige Software nicht mehr vollständig und im Originalzustand angeschaut werden. Das ist für die Langzeitarchivierung nicht akzeptabel, da nicht davon ausgegangen werden kann, dass diese proprietären Anwendungen für den Zeitraum der Archivierung zur Verfügung stehen. Daher sind offene standardisierte Datenformate für die Archivierung erforderlich. Im Folgenden werden zwei Beispiele dargestellt.

Im Rahmen des Open-Source-Projekts *OpenOffice* wurde das Standardformat *OpenDocument*[3] entwickelt. Dass der Hersteller der führenden proprietären Software im Bereich der Office-Anwendungen, Microsoft, sich dadurch genötigt sah, seinerseits ein offenes Format, OpenXML[4], zu entwickeln und zur Standardisierung einzureichen, kann nur als Reaktion auf den Erfolg von OpenDocument verstanden werden und nicht als Umdenken gegenüber Standards. Außerdem ist dieser Versuch, ein ursprünglich proprietäres Format offenzulegen und zu standardisieren, sehr umstritten.[5] Für die Konvertierung proprietärer Formate bietet sich die Standardisierung und Offenlegung des PDF-Formats in der Form des PDF/A (siehe ISO-19005-1:2005) an. Dieses stellt kein eigenständiges Format dar, sondern verwendet eine Teilmenge der in der PDF-Version 1.4 vorhandenen Elemente. Alle Elemente, die eine Lesbarkeit der Dokumente bei einer Langzeitaufbewahrung gefährden, wie z. B. Multimedia-Objekte innerhalb des Dokuments oder die Referenzierung auf nicht im Dokument enthaltene Fonts, werden nicht erlaubt.

3 Siehe http://www.oasis-open.org [4. Jan. 2008].
4 Siehe http://www.ecma-international.org [4. Jan. 2008].
5 Siehe http://www.odfalliance.org/resources/OfficeOpenXMLFactSheet.pdf.

Hardware

Die für die Archivierung verwendeten Speichersysteme sind keine Open-Source–Produkte. So stellen zum Beispiel *Content-Addressed-Storage-Systeme* proprietäre Produkte dar, da nur durch den Controller eines bestimmten Herstellers die Daten auf den Festplatten wieder adressiert und gelesen werden können. Ähnliches gilt auch für andere in der Archivierung verwendete Speichertechnologien.

Ein weiterer Vorteil von Open-Source-Software in Bezug auf die Hardware ist die Möglichkeit, durch Eingriffe in den Quelltext Anwendungen an neue Hardwareinfrastruktur anzupassen. Gerade modular gekapselte Speicherzugriffe können ohne Herstellerabhängigkeit für neue Hardwaretechnologien erweitert werden.

2.3 Beispiele für Open-Source-Anwendungen

Elektronisches Archiv – Fedora

Fedora[6] darf nicht verwechselt werden mit der Red-Hat-Linux-Distribution gleichen Namens. Fedora heißt *Flexible Extensible Digital Object and Repository Architecture* und wird inzwischen von einer eigenen neugegründeten *Non-Profit*-Organisation, den *Fedora Commons*, entwickelt.

Fedora liegt inzwischen in der Version 2.2.1 vor. Die erste Version wurde im Mai 2003 veröffentlicht. Die aktuelle Version bietet unter anderem folgende Funktionalitäten:

- Verbindung von digitalen Objekten mit *Web Services* (z. B. zur Bereitstellung der Objekte in unterschiedlichen Formaten *„on the fly"*)
- Versionierung einschließlich der Protokollierung von Veränderungen (*Audit Trail*)
- XML-Im- und -Export in den Formaten *Fedora Object XML* (FOXML) und *Metadata Encoding and Transmission Standards* (METS)
- Verknüpfung von Objekten als Tripel (Subjekt, Prädikat, Objekt)
- *Dublin Core*-Objektmetadaten

Als Schnittstellen werden zwei *SOAP*-basierte Webservices bereitgestellt, die Management- und die Access-API. In eingeschränkter Weise werden diese auch als HTTP-Service basierend auf *Representational State Transfer* bereitgestellt. Zusätzlich werden zwei Suchschnittstellen angeboten, eine für die einfache Suche in der relationalen Datenbank und eine für die Suche in der Kowari-Datenbank. Fedora unterstützt das *OAI Protocol for Metadata Harvesting* (OAI-PMH).

6 Siehe http://www.fedora-commons.org.

Elektronisches Archiv – DSpace

Die Open-Source-Software *DSpace*[7] ist eine gemeinsame Entwicklung des *Massachusetts Institute of Technologie* (MIT) und *Hewlett-Packard* (HP). Die *DSpace Foundation* wurde als Non-Profit-Organisation gegründet, um den weiten Anwenderkreis zu unterstützen. Das Ziel dieser Software ist die Erfassung, Speicherung, Indizierung, Aufbewahrung und Weitergabe von Forschungsmaterial und anderen Dokumenten im digitalen Format.

Die erste Version ist im November 2002 veröffentlicht worden. Die Software liegt zurzeit in der Version 1.4.2 vor. DSpace bietet folgende Funktionalitäten:

- Hierarchische Datenorganisation von Communitys bis Bitstreams
- Einfache (Dublin Core) und strukturelle Metadaten (Beziehung von Bitstreams untereinander)
- Import von einzelnen Dokumenten oder im Bündel in das System einschließlich Freigabeprozess
- *Global Unique Identifier* nach dem System der *Corporation for National Research Initiatives* (CNRI)
- Suchen und Blättern (Strukturierte und Volltextsuche)
- Export von Dokumenten und Metadaten
- Protokollierung von Ereignissen

Als Schnittstellen werden Java-Klassen auf drei verschiedenen Ebenen bereitgestellt, die aufeinander aufbauen: *Storage Layer*, *Business Logic Layer* und *Application Layer*. Während Fedora mehr ein *framework* bereitstellt, aus dem ein digitales Archiv mit entsprechender Benutzeroberfläche aufgebaut werden kann, bietet DSpace bereits eine einsatzbereite Implementierung einschließlich der Basis-Workflow-Funktionen. Ähnliche Open-Source-Anwendungen sind *EPrints*[8] und *CDS Invenio*[9].

Content und Records Management

Neben den Open-Source-Produkten, die dem universitären Bereich entstammen, gibt es inzwischen auch Open-Source-Software, die den Funktionsumfang für Enterprise–Content-Management-Produkte (ECM) hat. Mit ihr kann der gesamte Lebenszyklus von Informationsobjekten abgebildet werden. Ein Beispiel für die kommerzielle Orientierung dieser Art von Open-Source-Software ist *Alfresco*[10]. Die Software ist konzipiert von Entwicklern, die ursprünglich an proprietären ECM-Systemen gearbeitet

7 Siehe http://www.dspace.org.
8 Siehe http://www.eprints.org.
9 Siehe http://cdsware.cern.ch.
10 Siehe http://www.alfresco.com.

haben. Das Entwicklerteam des Herstellers Alfresco entwickelt das Kernprodukt. Der Quelltext wird veröffentlicht, um es der Alfresco-Community zu ermöglichen, das Produkt zu erweitern. Diese Erweiterungen können auch wieder in das Kernprodukt einfließen. Im Gegensatz zu den Archivsystemen aus dem universitären Bereich wird Alfresco kommerziell vermarktet. Der Funktionsumfang kann inzwischen mit proprietären ECM-Produkten verglichen werden.

Aber es gibt weitere Open-Source-Software, die zumindest Teilbereiche von ECM abdecken. Dazu gehören *Contineo*[11], *KnowledgeTree*[12] und *Xinco*[13].

Die Liste erhebt keineswegs den Anspruch auf Vollständigkeit. Vor allem die zahlreichen Open-Source-Web-Content-Management-Systeme sind hier nicht genannt, obwohl das *Web Content Management* auch zum ECM gehört. Traditionell sind Open-Source-Produkte im Bereich des Web Content Managements sogar sehr stark vertreten und gelten als sehr leistungsfähig. Ein Beispiel ist *Typo3*[14].

Harvester-Software

Eine besondere Art von Open-Source-Software stellt *Harvester-Software* dar, deren Aufgabe es ist, Objekte vornehmlich aus dem Internet zu sammeln und sie Archiven zur Verfügung zu stellen.

Die dänischen Archive der Staats- und Universitätsbibliothek und der Königlichen Bibliothek haben mit der *NetArchiveSuite*[15] ein solches Programm entwickelt, das wie ein *Web Crawler* das Internet durchforstet und das Material in so genannten ARC-Dateien für die Archivierung zur Verfügung stellt. Das Programm ist entwickelt worden, um das dänische Internet (Domäne .dk) zu archivieren. Die Daten werden nicht der allgemeinen Öffentlichkeit zur Verfügung gestellt. Daher stellen sich die im Abschnitt 2.1 beschriebenen Probleme mit dem Urheberrecht nicht.

Da es sich aber um Open-Source-Software handelt, kann das Programm auch von anderen Institutionen verwendet werden.

Die OAI-PMH-Schnittstelle ermöglicht es Service-Providern, Metadaten aus Archiven wie Fedora oder DSpace, die diese Schnittstelle unterstützen und als *data provider* agieren, zu sammeln und aufbereitet zur Verfügung zu stellen. Damit sind archivübergreifende Recherchen möglich.

3 Zusammenfassung und Ausblick

Manche sprechen von unserem Zeitalter als *dark digital age*. Durch den sorglosen Umgang mit digitaler Information und der rasenden Entwicklung im Bereich der

11 Siehe http://contineo.sourceforge.net.
12 Siehe http://www.knowledgetree.com.
13 Siehe http://www.xinco.org.
14 Siehe http://typo3.org.
15 Siehe http://netarchive.dk/suite.

digitalen Datenverarbeitung werden viele Informationen aus unserer Zeit nicht für nachkommende Generationen aufbewahrt, sondern gehen für immer verloren.

Wir vertrauen unsere digitalen Informationen proprietären Anwendungen und Datenformaten an, die von wenigen Unternehmen kontrolliert werden und deren Langzeitverfügbarkeit nicht gesichert ist. Die Lösung können nur offene Anwendungen und Datenformate sein. Dabei leistet die Open-Source-Community einen wesentlichen Beitrag, nicht zuletzt deshalb, weil sie nicht mehr nur als *Modeerscheinung* betrachtet wird, sondern als zuverlässige und einsetzbare Alternative zu proprietärer Software.

Open-Source-Software allein ist aber nicht die Lösung für die Herausforderung der Bewahrung digitaler Information. Einen wichtigen Beitrag leistet auch Open Content, der ungehindert von Urheberrecht und Copyright archiviert werden kann. Unabhängig von Open-Source- oder proprietärer Software ist es für die Langzeitverfügbarkeit digitaler Information notwendig, dass Verantwortung für die Archivierung übernommen wird. Diese Archivierung muss als Aufgabe verstanden werden, die nicht nur rechtlich, sondern auch kulturell und gesellschaftlich gefordert ist. Hier muss sich die Open-Source-Community fragen lassen, wie sie mit ihren Inhalten bezüglich der Archivierung umgeht. Wer trägt hier die Verantwortung für die Langzeitaufbewahrung digitaler Informationen?

Wie sehr insgesamt die Erkenntnis wächst, welche Aufgaben uns das digitale Zeitalter hinsichtlich der Bewahrung von Information stellt, zeigen nationale und internationale Projekte wie *NESTOR*[16] und *PLANETS*[17], die sich sicherlich auch der Vorteile von Open Source bedienen werden.

Literatur

Heuer, S. (2006), Ein Archiv für die ganze Welt, *in* B. Lutterbeck, M. Bärwolff und R. A. Gehring (Hrsg.), 'Open Source Jahrbuch 2006 – Zwischen Freier Software und Gesellschaftsmodell', Lehmanns Media, Berlin, S. 283–296.

Kampffmeyer, U. (2004), 'Elektronische Archivierung und Storage-Technologien'. **URL:** *http://www.ecmguide.de/magazin/archivierung.asp?todo=de&theID=396&bhcp=1*

Kampffmeyer, U. (2005*a*), 'GDPdU und elektronische Archivierung', *PROJECT CONSULT Newsletter* (20050531), S. 20 ff.

Kampffmeyer, U. (2005*b*), 'Herausforderung E-Mail-Archivierung', *PROJECT CONSULT Newsletter* (20051014), S. 26 ff.

Kampffmeyer, U. (2007*a*), 'Information Management Compliance'. KoelnMesse.

Kampffmeyer, U. (2007*b*), 'MoReq Update', *PROJECT CONSULT Newsletter* (20070309), S. 19 ff.

16 Siehe http://www.langzeitarchivierung.de/index.php.

17 Siehe http://www.planets-project.eu.

Kampffmeyer, U. und Rogalla, J. (1997), 'Grundsätze der elektronischen Archivierung', *VOI-Kompendium* **3**.

Spindler, G. (2006*a*), 'Rechtliche Rahmenbedingungen von Open Access-Publikationen', *Göttinger Schriften zur Internetforschung* **2**.

Linux4Afrika – Ein Linux-Terminal-Server-Projekt

HANS-PETER MERKEL

Das Projekt *Linux4Afrika* hat sich zum Ziel gesetzt, afrikanischen Schulen und deren Schülern besseren Zugang zur Informationstechnologie zu verschaffen. In den Jahren 2006 und 2007 stattete der Trägerverein FreiOSS daher Schulen in Mosambik und Tansania mit in Deutschland gespendeten gebrauchten Computer aus. Der Autor schildert die Erfahrungen, Probleme und Herausforderungen, mit denen das Projekt bisher in Berührung gekommen ist.

Schlüsselwörter: Afrika · Terminalserver · Entwicklungsprojekt

1 Einleitung

Seit vielen Jahren wird immer wieder versucht, den schlechten Zugang zur Informationstechnologie in vielen afrikanischen Ländern ins Bewusstsein der Öffentlichkeit zu rücken. Otter (2006, S. 382 ff.) schreibt, dass „[i]m Jahr 2004 weniger als drei von hundert Afrikanern Internetzugang [hatten], verglichen mit durchschnittlich einem von zwei Bewohnern der G8-Länder". In den afrikanischen Schulen sei die Situation teilweise noch dramatischer, so mussten sich beispielsweise in Namibias Schulen im Jahr 2004 durchschnittlich acht Lehrer und 280 Schüler einen einzigen Computer teilen, wobei grundlegende Anforderungen an die Computernutzung, wie eine kontinuierliche Stromversorgung, keineswegs sichergestellt seien. Dieser Artikel beschreibt eine der vielen kleineren Initiativen, die versuchen, die Lage in Afrika zu verbessern und geht dabei besonders auf die vielfältigen und oft unerwarteten Schwierigkeiten und Hindernisse ein, die ein solches Unterfangen mit sich bringt.

Seit 2004 ist das *Freiburger Open-Source-Netzwerk*[1] (FreiOSS) als gemeinnütziger Verein mit mehr als 300 Mitgliedern aus 33 Ländern tätig. FreiOSS unterstützt und fördert den Einsatz von Open Source auf nationaler und internationaler Ebene. Die

1 Siehe http://www.freioss.net.

Mitglieder des Vereins kommen überwiegend aus der IT-Branche und viele bringen Erfahrung in der technischen Entwicklungszusammenarbeit mit.

In den vergangenen Jahren waren einige der Vereinsmitglieder für ein IT-Entwicklungshilfeprojekt der *InWEnt gGmbH*[2] tätig. Die Teilnehmer des Projekts waren IT-Experten aus asiatischen und afrikanischen Ländern. Sie durchliefen ein einjähriges Trainingsprogramm in Deutschland mit dem Titel *it@ab – Information Technology in African Business*.[3] Während dieses Trainings absolvierten mehrere Teilnehmer aus Afrika ein dreimonatiges IT-Praktikum am *St. Ursula-Gymnasium* in Freiburg, wo der Verein früh eine Thin-Client-Architektur aufbauen konnte, und betreuten das dort eingesetzte *Linux-Terminal-Server-Projekt*[4] (LTSP). Während der Praktika–Zeiten wurde von den afrikanischen Studenten die Frage aufgeworfen, ob es nicht möglich sei, eine solche LTSP-Lösung für ihre Heimatländer zu adaptieren.

Im Frühjahr 2006 wurden dann im Vorstand des FreiOSS die Möglichkeiten, Bedingungen und Chancen eines LTSP-Projekts für Afrika besprochen. Dabei wurde beschlossen, zunächst in Mosambik und Tansania mit einem Projekt für jeweils eine Schule zu starten. Der Hauptgrund für die Wahl dieser beiden Länder war, dass der Verein sowohl in Mosambik als auch in Tansania über engagierte Mitglieder mit gutem IT-Know-how verfügte. Die lokale Unterstützung des Projekts war für FreiOSS sehr wichtig, um die Installation und die weitere kontinuierliche Betreuung der Schulen vor Ort zu gewährleisten und somit für eine nachhaltige Entwicklung des Projekts zu sorgen. Das Projekt *Linux4Afrika* war geboren. Das Ziel war, zunächst 40 gebrauchte Computer als Thin-Clients und zwei Server zu sammeln. In den folgenden Monaten wurden alle Mitglieder über das Projekt informiert und um Mithilfe gebeten. Außerdem startete die Internetseite linux4afrika.de, um die Entwicklung des Projekts zu dokumentieren.

2 Erste Schritte in Mosambik

Im Vorfeld eines im Juni 2006 geplanten Workshops zum Thema in Mosambik bat FreiOSS die dortigen Mitglieder, eine geeignete Schule für das LTSP-Projekt in Maputo zu suchen. Von FreiOSS gespendete *Barebones* sollten im Rahmen des Workshops an einer Schule installiert werden, um damit den Grundstein für einen Computer-Pool mit LTSP zu legen. Die Suche nach einer geeigneten Schule gestaltete sich nicht einfach. Problematisch erwies sich z. B., dass Schulmitarbeiter teilweise nicht bereit waren, ihre Freizeit für die Unterstützung des Projekts zur Verfügung zu stellen. Kurz vor dem Workshop berichteten Vereinsmitglieder aus Mosambik aber, dass man doch eine geeignete Schule gefunden habe, die sehr an Linux und dem Projekt interessiert sei. Leider stellte sich beim Übergabetermin heraus, dass die Schule bereits über

2 Siehe http://www.inwent.org und http://www.it-inwent.org [10. Feb. 2008].
3 Siehe http://www.it-ab.org [10. Feb. 2008].
4 Für weitere Informationen siehe http://www.ltsp.org/ [10. Feb. 2008].

einen recht passablen Pool mit Linux-Computern verfügte. Dennoch war man hier natürlich sehr dankbar über die Computerspende, da es immer noch an Arbeitsplätzen in ausreichender Zahl mangelte. Für FreiOSS jedoch war die Auswahl dieser Schule letztlich weniger befriedigend. Der Verein hätte eher eine Schule vorgezogen, die noch über keinen vergleichbaren Pool verfügt.

Im weiteren Verlauf wurde dieses erste Projekt in Mosambik zugunsten des im Folgenden beschriebenen Projekts in Tansania zurückgestellt. Das lag hauptsächlich daran, dass FreiOSS in Mosambik zum damaligen Zeitpunkt nur noch ein Mitglied hatte, das sich aktiv für *Linux4Afrika* engagierte. Dies war für ein Projekt, das für die Auswahl der Schulen, die Installation der Computer und die weitere Betreuung verlässliche Partner vor Ort benötigte, eindeutig zu wenig. Außerdem blieb die Frage offen, wie der Transport nach Mosambik finanziert werden sollte. Der Vorstand von FreiOSS hatte sich darauf verständigt, einen Eigenanteil der Länder etwa bei den Transportkosten anzustreben: entweder durch eine Finanzierung von staatlicher Seite oder durch private Sponsoren.

3 Das Hauptprojekt in Tansania

Die Arbeit in Tansania begann im August 2006, als der Autor auf Einladung von Paul Koyi in Bagamoyo/Tansania ein Open-Source-Training durchführte. Paul Koyi ist CEO von *Agumba Computer Ltd.*, einem Computerdienstleister in Daressalam und Mitglied bei FreiOSS. Über einen Teilnehmer des Trainings, der von dem Linux4Afrika-Projekt erfahren hatte, wurde ein Kontakt mit einem Minister der Regierung Tansanias geknüpft. Bei einem Besuch in Bagamoyo während des Trainings informierte dieser sich über den Verein und das Projekt. Er zeigte sich von der Idee begeistert und fragte an, ob FreiOSS in der Lage wäre, 100 Computer zu sammeln und diese Ende des Jahres 2007 nach Tansania zu verschicken. Sein Ziel war, in seinem Wahlkreis damit fünf Schulen mit der LTSP-Lösung auszustatten. Als Eigenbeitrag stellte er die Übernahme der Transportkosten für den Computercontainer durch die Regierung in Aussicht.

Paul Koyi hatte zum damaligen Zeitpunkt bereits angeboten, als Projektpartner für *Linux4Afrika* in Tansania zu arbeiten und mit den Mitarbeitern seiner Firma für die Installation der Computer, die Ausbildung der Lehrer und für die weitere Betreuung vor Ort zu sorgen. Im Rahmen der InWEnt-Trainings durchliefen mehrere Mitarbeiter seiner Firma, ebenfalls Mitglieder von FreiOSS, in Deutschland bereits ein Linux-Training, waren also mit OSS gut vertraut. Der Verein nahm dieses Angebot gerne an.

Da mit einer Hardwaresammlung in größerem Stil erst begonnen werden sollte, nachdem die Übernahme der Transportkosten durch Tansania gesichert sein würde, nahm FreiOSS Ende 2006 Kontakt mit dem Minister auf, um die Einzelheiten des Transports zu besprechen. Dieser verwies an die tansanische Botschaft in Berlin. Sie

würde sich „um alles kümmern". Die Botschaft setzte sich mit dem Verein in Verbindung und zeigte sich sehr interessiert an dem Projekt. In den nächsten Wochen holte die Botschaft Angebote bei Speditionen für den Transport ein und es schien alles bestens geregelt. Doch die Spedition, die letztendlich von der tansanischen Botschaft telefonisch den Zuschlag für den Transport erhielt, bat dann FreiOSS, den schriftlichen Auftrag zu unterschreiben, was mit Verweis auf die Botschaft als Auftraggeber abgelehnt wurde. Über Wochen herrschte Funkstille und die Projektmitarbeiter bezweifelten, ob die Finanzierung der Kosten für den Container über die Regierung zustande kommen würde. Daher begann man, nach alternativen Lösungen zu suchen.

Zwischenzeitlich wurde die Finanzierung des Transports wie in einem ähnlichen Projekt für Nepal angestrebt, bei dem die *Deutsche Gesellschaft für Technische Zusammenarbeit* (GTZ) die Transportkosten übernommen hatte.[5] Nach Vorstellung des Linux4Afrika-Projekts bei der GTZ in Tansania bekam FreiOSS aber leider einen ablehnenden Bescheid, da die GTZ zur damaligen Zeit keine Möglichkeiten hatte, das Projekt zu unterstützen. Gelöst wurde das Problem der Transportfinanzierung schließlich durch das Angebot unseres Partners Paul Koyi, der die vollständigen Transportkosten für den Container übernehmen wollte. Diese Lösung ergab sich allerdings erst sehr spät, weswegen während der Hardwaresammlung diesbezüglich viel Unsicherheit herrschte.

Im Frühjahr 2007 lief das Projekt auf Hochtouren. Das anvisierte Ziel von 100 Computern war Ende 2006 bereits erreicht und nach dem *LinuxTag* im Juni waren beinahe 200 Computer gesammelt. Es schien, dass das Thema „Afrika" im Vorfeld des G8-Gipfels in Heiligendamm wieder mehr ins öffentliche Bewusstsein rückte. So sendete *3sat* nach dem *LinuxTag* drei Beiträge über *Linux4Afrika* und viele lokale Zeitungen berichteten. Diese Medienauftritte sorgten dafür, dass FreiOSS Angebote für Hardwarespenden aus ganz Deutschland bekam.

Da sich durch die ungeklärte Transportkostenfrage der Termin für das Verschicken des Containers immer weiter nach hinten verschob, nahm auch der gespendete Hardwarebestand ständig zu. Sorge machte der Lagerplatz für die Gerätschaften. Zunächst wurden die Spenden auf Kellerräume und Garagen von Mitgliedern verteilt, was jedoch schnell sehr unübersichtlich und arbeitsaufwändig wurde. Gegen Ende der Sammlung bekam der Verein dann zum Glück die Erlaubnis, den EDV-Raum der Volkshochschule in March als Lager zu nutzen. Ein idealer Standort, da die Computer hier auch getestet werden konnten.

Der Container sollte spätestens Ende Juli vor Ferienbeginn in Baden-Württemberg gepackt und verschickt werden. Es wurden nun eigene Angebote von ortsansässigen Speditionen eingeholt und im Juni eine Spedition mit dem Transport beauftragt. Am 14. Juli 2007 wurde dann ein Container mit 202 Thin-Clients, 10 Servern und 150 Monitoren sowie weiteren Hardware-Komponenten (Switches, USV, . . .) bepackt.

5 Siehe „Ganesha's Project" unter http://www.bpb.de/methodik/HC7S2V,0,0,Freie_Software_in_Entwicklungsländern_echte_Hilfe_zur_Selbsthilfe.html [10. Feb. 2008].

Dieser sollte ursprünglich am 15. August 2007 in Daressalam ankommen, konnte aber wegen fehlender Zollpapiere in Rotterdam erst am 4. August 2007 verschifft werden.

4 Verwendete Software und Konfiguration

Die für *Linux4Afrika* zunächst erwogene Lösung mit *Debian* und LTSP wurde zugunsten einer Edubuntu-Lösung verworfen, da diese einfacher zu handhaben ist. LTSP 5 ist auf *Edubuntu* vorkonfiguriert und erfordert nur geringe EDV-Kenntnisse des Administrators. Gerade in Afrika, wo die zuständigen Lehrkräfte zwar mit großem Engagement, aber oftmals wenig oder gar keinem Vorwissen die Wartung der Computerpools übernehmen müssen, stellt dies eine große Arbeitserleichterung dar.

Schulen außerhalb von Daressalam oder Maputo werden bei auftretenden Problemen zunächst auf sich alleine gestellt sein und müssen versuchen, diese in Eigenregie zu lösen. Kommt es z. B. zu einem Ausfall, bei dem die Systemfestplatte betroffen ist, können die Schulen auf eine Kopie der Serverfestplatte zurückgreifen, welche jede Schule ausgehändigt bekommt. Auf aufwändige RAID-Systeme wurde dabei bewusst verzichtet, da der Austausch einer normalen Festplatte auch mit der einfachen Ausbildung der zuständigen Lehrer vor Ort durchgeführt werden kann. Aber auch bei einem Serverausfall kann noch weiter an den Thin-Clients gearbeitet werden: *Ubuntu* spendete freundlicherweise 500 Live-CDs, die es ermöglichen, lokal vom Thin-Client per CD zu booten und mit der Ubuntu-Live-CD zu arbeiten, wofür kein Server benötigt wird.

Serverseitig wurden der Afrika-Musterlösung noch einige Dienste hinzugefügt. Ein LAMP-Server mit dem Offlineauftritt von *Linux4Afrika* bietet den Schulen die Möglichkeit, die Entstehung des Projekts lokal nachzuvollziehen. Mit der OSS-Lernplattform *Moodle*, einem weiteren webbasierten Projekt, erhalten sowohl Lehrer als auch Schüler die Möglichkeit, Lernmaterial offline abzurufen und selbst einzustellen. Hier ist auch unsere Hard- und Softwaredokumentation abrufbar. Außerdem wurden ein Mailserver und zusätzlich *Postfix* und ein POP3-Server installiert, wobei *Mozilla Thunderbird* als Mail-Client dient. Die Schüler mit Techniken wie E-Mail vertraut zu machen, ist auch an Schulen ohne Onlinezugang wichtig, um den Anforderungen, die junge Menschen zukünftig in einer globalisierten Welt erwarten, gerecht zu werden. Eingebunden wurde auch die Offline-Wikipedia *Kiwix*.[6]

Für die Terminalserver in Tansania wurde zunächst über die Installation von Kisuaheli-Software-Versionen nachgedacht. *Mozilla* (Version 1) und *OpenOffice.org* sind auch in der Landessprache erhältlich. FreiOSS-Mitglieder, die diese Versionen in Tansania testeten, waren aber eher unzufrieden, da viele Menüpunkte unglücklich übersetzt sind und eher für Verwirrung sorgen. Deshalb wurde doch die englische Version gewählt.

6 Siehe http://www.kiwix.org/index.php/Main_Page [10. Feb. 2008].

Trotz sehr geringem Wartungsaufwand benötigen auch Terminalserver administrative Betreuung. Nach unseren Erfahrungen müssen hier vor allem zwei Dinge gepflegt werden: die Benutzerverzeichnisse und die Logdateien. Während die Logdateien automatisch wöchentlich gelöscht werden (ohne Internetzugang bedeutet dies kaum eine Einschränkung), werden die Verzeichnisse der Benutzer bei Bedarf manuell gelöscht. Jeder Terminalserver besitzt zusätzlich zum Benutzerkonto des Systemverwalters 25 Benutzerkonten (tsuser1 – tsuser25). Diese User haben Quotabeschränkungen für ihre Verzeichnisse, um die Festplatte nicht zum Überlaufen zu bringen.

Hardware	Verwendung
Grafikkarten und Monitore	Alle Thin-Clients müssen unabhängig von ihrer Grafikkarte eine Grafikauflösung von $1024 * 768$ liefern (VESA-Standard). Eine Anforderung, die damit automatisch auch für alle Monitore gilt. Außerdem werden AGP-Grafikkarten vorausgesetzt, da PCI-Karten meist zu alt sind und nicht genügend Speicher haben. Daraus ergibt sich ein PII-System als Minimalanforderung.
Arbeitsspeicher	Ein „Flaschenhals" sind die Speichermodule. Die durchgeführten Tests ergaben, dass eine sinnvolle Systemstartzeit erst ab 64 MB Arbeitsspeicher erreicht wird. Durch großzügige Spenden konnten die Thin-Clients mit 96-128 MB Arbeitsspeicher ausgestattet werden.
Festplatte	Alle Festplatten werden aus den Clients ausgebaut. Sind sie groß genug, werden darauf Serverkopien erstellt. Alte Platten werden entsorgt, Anschlusskabel entfernt. Damit ist es auch nicht mehr möglich, dass der Thin-Client mit einer (Raub-)Kopie eines Microsoft-Betriebssystems ausgestattet wird.
Laufwerke	Diskettenlaufwerke und/oder CD/DVD-Laufwerke verbleiben in den Thin-Clients und sind mittels *Local Device Support* nutzbar.
Tastaturen	In Tansania werden entweder britische oder US-amerikanische Tastaturen eingesetzt. Diese sind in Deutschland schwer zu beschaffen. Eine größere Menge englischer Tastaturen besorgte eine engagierte Studentin aus Mannheim über die dortige Universität. Ansonsten werden deutsche Tastaturen mitgeliefert. Um eine spätere Tastaturumstellung einfach durchzuführen, werden zwei Keyboardlayouts installiert. Per Mausklick kann die Einstellung vorübergehend oder in den Systemeinstellungen dauerhaft geändert werden. Durch die portugiesische Landessprache fallen in Mosambik ebenfalls Änderungen der Spracheinstellung auf den Servern an.

Tabelle 1: Verwendete Hardware des Linux4Afrika-Projekts.

Diese Begrenzungen reichen aber nicht aus. Die Benutzer verstellen oft aus Unwissenheit ihre persönlichen Einstellungen oder ändern die Standardpasswörter. Die Folge ist, dass diese Benutzerkonten für andere Schüler nicht mehr nutzbar sind. Um dies zu verhindern, wurden zwei Kommandozeilenskripte installiert. Das erste entfernt die Logdateien, das zweite entfernt alle Benutzer und ihre Verzeichnisse und

legt sie mit neuen Profilen und Standardpasswörtern wieder neu an. Da dieser Schritt jedoch alle eigenen gespeicherten Dokumente löscht, besteht durch den *Local Device Support* von LTSP 5 die Möglichkeit, selbst erstellte Dokumente auf einem Memory–Stick zu speichern. So kann in der nächsten Unterrichtsstunde ein anderer Schüler an dem Computer arbeiten. Durch den gleichen Loginnamen hat dieser Zugriff auf die Daten des vorherigen Benutzers, die auf dem Server gespeichert sind.

5 Verwendete Hardware

Vor dem Transport wird jeder einzelne Thin-Client geprüft und mit *Edubuntu* gebootet. Auch jeder Monitor wird auf Tauglichkeit getestet. Damit wird sichergestellt, dass nur funktionierende Hardware verschickt wird. Jedoch hängt der erfolgreiche Computerbetrieb in Tansania nicht nur von der Funktionstüchtigkeit der Geräte ab.

Gerät	Verwendung
Netzwerkkarte	Der größte Teil der gespendeten Thin-Clients verfügt über normale Netzwerkkarten. Nur wenige bringen die Voraussetzung zum Netzwerkbooten per *PXE* mit. Die meisten Unternehmen, die Hardware spenden, haben Netzwerkkarten übrig, da die neuen Computer meist über Onboard-Netzwerkanschlüsse verfügen. Handelsübliche Netzwerkarten wie *3C90x*, *RTL 8139* oder *Intel 100 Pro* sind deshalb häufig eine Gratisbeigabe. *3C905C* und *Intel 100 Pro* verfügen bereits über *PXE*, die anderen Netzwerkkarten haben oft einen leeren Sockel zum Nachrüsten.
Server	Für das Ziel, ca. 20 Clients an einen Server anzubinden, benötigt dieser mindestens 2 GB Arbeitsspeicher. Alle gebrauchten Server nutzten *SD-* oder *DDR-RAM*. Eine Aufrüstung auf 2 GB ist teurer als die Anschaffung eines neuen Mainboards. Daher wurde beschlossen, neue Mainboards zu kaufen. Für die *AMD64-CPU* werden preiswerte Mainboards (Sockel *AM2*) angeboten. Mit 2 GB *DDR2-RAM* betragen die derzeitigen Kosten für Board, CPU und RAM knapp 300 €.

Tabelle 2: Netzwerkinfrastruktur des Linux4Afrika-Projekts.

Eines der großen Probleme Tansanias ist derzeit die instabile Stromversorgung. Tansania bezieht rund 86,2 Prozent seines Stroms aus Wasserkraft.[7] Insbesondere in der Trockenzeit kommt es durch leere Stauseen zu Engpässen. Während der IT–Trainings gab es pro Stunde durchschnittlich fünf kurze Stromausfälle, die manchmal mehrere Minuten anhielten. Dies ist in Tansania ein durchaus alltägliches Ereignis, deshalb sind die meisten Computer mit unterbrechungsfreien Stromversorgungen (USV) gepuffert.

Hier kommt ein wichtiger Vorteil der Thin-Client-Technologie zum Tragen: Die Thin-Clients können entweder ganz ohne USV betrieben oder aufgrund des gerin-

7 Siehe http://www.suedwind-institut.de/downloads/Wasserkraft-Inga_deutsch.pdf [10. Feb. 2008].

gen Stromverbrauchs zu mehreren Geräten an einer USV zusammengefasst werden. Kommt es jedoch wie im Januar 2006 zu Stromrationierungen von bis zu acht Stunden pro Tag, ist ein Schulbetrieb an Computern natürlich nicht mehr möglich.

Da Thin-Clients, wie sie heute erhältlich sind, mit knapp 100 € pro Gerät für das Linux4Afrika-Projekt nicht erschwinglich sind, bleiben nur die schon beschriebenen Gerätespenden. Dabei kam dem Projekt zugute, dass viele Unternehmen derzeit ihre 500- beziehungsweise 800-MHz-Computer aussondern, die dazu oft in handlicher Barebone-Größe sind.

6 Fazit und Ausblick

Inzwischen, Mitte September 2007, ist der erste Container in Tansania angekommen und wurde vor Ort entladen. Das Haus, in dem die Geräte in Daressalam lagern, wird rund um die Uhr von vier Wachleuten bewacht. Zwischenzeitlich ist die erste Schule mit gespendeten Computern ausgestattet. Ein Journalist von *Linux New Media* war bei der Übergabe vor Ort und berichtete in einer Ausgabe des *Linux Magazins* darüber.[8] Von August bis November 2007 waren bei FreiOSS zwei Praktikanten aus Tansania zu Gast, die während ihres Praktikums ein intensives IT-Training absolvierten. Hierbei konnten sie sich gute Kenntnisse mit den für sie neuen OSS-Produkten aneignen und werden nun in Tansania in der Anfangsphase des Projekts zusätzlich als Partner den Schulen zur Seite stehen und ihnen beim Aufbau der Computer-Pools und der Ausbildung des Personals helfen. Ermöglicht wurde dieser Praktikumsaufenthalt wiederum durch Paul Koyi, der für die Kosten der Flüge, Unterkunft und die Verpflegung der beiden aufkam. Mittlerweile wurde auch mit der Hardwaresammlung für einen zweiten Container begonnen. Ende des Jahres 2007 sollte er nach Tansania verschickt werden. Dieser Termin konnte allerdings nicht eingehalten werden, da es wieder Probleme mit der Finanzierung des Transports gab. Ein interessierter Sponsor, *Easy Finance* aus Tansania, machte einen Rückzieher, nachdem man die Computer der letzten Lieferung begutachtet hatte. FreiOSS wurde per Mail mitgeteilt, dass man nicht beeindruckt von den Geräten sei, da diese zu alt seien. Es sollten doch bitte neuere Computer geschickt werden, dann würde der Finanzierung nichts im Wege stehen. Hier muss noch einiges an Aufklärungsarbeit in Bezug auf LTSP geleistet werden. Eine Frage, die sich in diesem Zusammenhang stellt, ist, ob ein Teil der Kosten über den Versand von gebrauchten „Stand-Alone-Computern", die dann auch mit einem MS-Betriebssystem laufen und in Afrika als Second-Hand-Computer verkauft werden könnten, gedeckt werden sollte. Dies ist eine Methode, die einige andere Hilfsprojekte auch anwenden, die jedoch innerhalb des Vereins sowohl Befürworter als auch Gegner hat, weshalb es noch keine Entscheidung gab.

Im Februar 2008 werden einige FreiOSS-Mitglieder nach Tansania reisen. Dann werden die Computer bei einer offiziellen Übergabefeier in Betrieb genommen und

8 Siehe http://www.linux-magazin.de/heft_abo/ausgaben/2007/12/im_land_der_gnus [10. Feb. 2008].

das Projekt kann vor Ort begutachtet werden. Außerdem ist eine Infoveranstaltung für Schulleiter und Lehrer über Open Source und unser LTSP-Projekt geplant. Des Weiteren sollen auch Workshops für Lehrer und Schüler abgehalten werden, um sie mit *Ubuntu/Edubuntu* vertraut zu machen.

Im März 2008 wird *Linux4Afrika* dann auf der *CeBIT* vorgestellt. Diese Messeauftritte wie *LinuxTag* oder *Ubucon* sind wichtig, da nach den Messeauftritten die Zugriffe der Projekt-Webseite sprunghaft ansteigen und viele Hilfsangebote per Mail hereinkommen. Zum *LinuxTag 2007* wollte uns unser tansanischer Partner auf der Messe unterstützen. Das Vorhaben scheiterte jedoch daran, dass er erst einige Stunden vor seinem gebuchten Abflug von der deutschen Botschaft sein Visum erhielt. Zur *CeBIT 2008* hat er erneut seine Beteiligung und Hilfe angeboten, was hoffentlich nicht wieder an bürokratischen Barrieren scheitert.

Viele Fragen sind noch offen: Wie werden die Umsetzungsschritte in Tansania ablaufen? Wie wird die Akzeptanz des Open-Source-Projekts bei Lehrern und Schülern sein? Fragen, deren Antworten wichtig für die Gesamtevaluierung des Projekts sind. Erfolgreich war mit Sicherheit der bisherige Ablauf des Projekts hier in Deutschland. Trotz einiger Probleme wurde doch vieles erreicht. Es wurde viel Aufklärungsarbeit in Sachen Open Source geleistet und viele neue Mitstreiter würden für das Projekt gewonnen.

Literatur

Otter, A. (2006), Digitale Möglichkeiten für Afrika, *in* B. Lutterbeck, M. Bärwolff und R. A. Gehring (Hrsg.), 'Open Source Jahrbuch 2006 – Zwischen Softwareentwicklung und Gesellschaftsmodell', Lehmanns Media, Berlin, S. 381–387.

Glossar

ACL (Access Control Lists, Zugriffskontrolllisten) werden zur Regelung von Zugriffsberechtigungen auf Ressourcen in IT-Systemen verwendet. Mit den einzelnen Einträgen in einer ACL kann für ein Objekt im System detailliert festgelegt werden, für welche Benutzer, Benutzergruppen oder Prozesse welche Art von Zugriff verboten oder erlaubt wird. · **Active Directory** ist ein zentraler Verzeichnisdienst zur Verwaltung eines Netzwerkes – hierzu zählen Benutzer- und Gruppenkennzeichen, Arbeitsplatzcomputer, Server und Drucker. · **Apache** ist ein weit verbreiteter Open-Source-Webserver, der ursprünglich aus dem NCSA-Webserver entstanden ist, unter der *Apache License* steht und federführend von der *Apache Software Foundation* betrieben wird. · **API** (Application Programming Interface – dt. Programmierschnittstelle) ist eine Menge von definierten Spezifikationen, die dem Programmierer den Zugriff auf Funktionen einer Software ermöglichen. · **ASP** (Application Service Providing) bezeichnet das Anbieten von Anwendungen, die Kunden eines Application Service Providers z. B. über das Internet zur Verfügung gestellt werden. Der Application Service Provider kümmert sich um die gesamte Administration, wie Datensicherung, das Einspielen von Patches usw. · **ATOM** (Atom Syndication Format) ist ein XML-Format für den plattformunabhängigen Austausch von Informationen. Ähnlich einem Nachrichtenticker kann ein geeignetes Programm, welches dieses Format unterstützt, von einer vom Benutzer gewählten Quelle Informationen empfangen und anzeigen. · **B2C** (Business-to-Consumer) beschreibt Kommunikations- und Handelsbeziehungen zwischen Unternehmen und Privatpersonen, im Gegensatz zu Beziehungen zwischen Unternehmen oder Behörden (B2B). · **Backend** ist das Gegenstück zum Frontend, z. B. bei verteilten Anwendungen. Während das Frontend den für den Benutzer sichtbaren Teil der Anwendung darstellt, ist das Backend der für ihn unsichtbare, quasi im Hintergrund arbeitende Teil. Ein Bankautomat beispielsweise, an dem der Kunde Geld abhebt, stellt ein Frontend dar und der zentrale Server, der das Kundenkonto verwaltet, das Backend. · **Barebone** ist die Bezeichnung für eine bestimmte Computerkonfiguration. Als Barebone bezeichnet man Computer, die (meist beim Verkauf) nur aus Gehäuse, Netzteil, Prozesserkühler und Hauptplatine bestehen. Die bei diesen Konfigurationen meist verwendete kleinere Gehäusebauform wird selbst gelegentlich als Barebone bezeichnet. · **Berliner Erklärung** ist die Kurzform für die „Berliner Erklärung über offenen Zugang zu wissenschaftlichem Wissen", die im Oktober 2003 auf einer Tagung der Max-Planck-Gesellschaft von deutschen und internationalen Forschungsorganisationen unterzeichnet worden ist. Sie ist ein wichtiger Meilenstein der Open–Access-Bewegung. · **Best Practice** ist ein Prozess oder eine Vorgehensweise, der/die sich in der Praxis mehrfach bewährt hat. · **Blackboard-System** dient in der künstlichen Intelligenz (KI) der Kommunikation der Programmstrukturen eines KI-Systems untereinander, indem es ihnen einen gemeinsam nutzbaren Speicherraum zur Verfügung stellt. · **Blog** ist die Kurzform von *Weblog*, einem öffentlichen Online-Journal, welches Beiträge chronologisch auflistet

und die Möglichkeit bietet, die Einträge zu kommentieren und mit anderen Blogs zu verlinken. · **BSD-Lizenz** (Berkeley Software Distribution License) ist eine von der *Open Source Initiative (OSI)* anerkannte Open-Source-Lizenz, die im Gegensatz zur *GPL* die Weitergabe von Software auch ohne den Quelltext billigt. Dadurch ist sie weniger restriktiv als die *GPL*. · **Bug-Tracker** bezeichnet ein System, zur Entdeckung, Verfolgung und Beseitigung von Programmfehlern. · **Business Intelligence** ist der Prozess, interne und externe Geschäftsinformationen zu sammeln und zu analysieren. Dies soll die Entscheidungsfindung der Leitungsebene unterstützen. · **Card Sorting** ist eine Technik um zu erforschen, wie Menschen bestimmte Gegenstände, Begriffe oder Einheiten gruppieren, damit Strukturen entwickelt werden können, welche die Wahrscheinlichkeit maximieren, Nutzer bestimmte Elemente finden zu lassen. Es ist einfach und billig durchführbar, ermöglicht herauszufinden, wie „echte Leute" Gruppierungen vornehmen würden, hilft beim Auffinden von schwer kategorisierbaren Objekten und identifiziert Technologien, die leicht missverstanden werden können. · **CEO** steht für *Chief Executive Officer*, in Deutschland der Unternehmensleiter, Geschäftsführer oder Vorstand. · **Cluster** ist ein Netzwerk von Computern, das nach außen hin wie ein einzelner Rechner agiert. Die interne parallele Bearbeitung von Aufgaben führt zu einer deutlich höheren Leistung. Cluster sind meist so organisiert, dass der Ausfall einzelner Computer nicht den Ausfall des gesamten Clusters nach sich zieht. · **CMS** (Content-Management-System) ist eine Applikation, die gemeinschaftliches Erstellen und Bearbeiten von elektronischen Dokumenten (Text, Multimedia) ermöglicht und organisiert. Wert gelegt wird auf eine möglichst einfache Bedienung, die keinerlei Programmierkenntnisse erfordert und z. B. ein schnelles Publizieren auf Webseiten erlaubt. · **Code-Review** ist ein iterativer Prozess, in dem geschriebener Quelltext von verschiedenen Personen validiert und auf Fehler hin untersucht wird. Dies soll eine hohe Qualität der Software gewährleisten. · **Colocation** bezeichnet das Anmieten von Raum in einem Rechenzentrum, um dort eigene Server aufzustellen, und von der vorhandenen Infrastruktur (z. B. Netzbandbreite) zu profitieren. · **Community** bezeichnet im Allgemeinen eine Gruppe von Individuen, die durch ein gemeinsames Element – meist sozialer Natur – verbunden sind, etwa ein gemeinsames Ziel oder eine gemeinsame Aufgabe. · **Content-Management-System** siehe CMS. · **Copyleft** bezeichnet eine Methode, die Freiheit der Nutzung, Modifikation und Weiterverbreitung einer Software zu sichern. So legt die *GPL* beispielsweise unter Verwendung des Konstrukts des Urheberrechts eines Autors fest, dass sämtliche Nutzungen und Modifikationen der Software erlaubt sind, eine Weiterverbreitung aber nur dann, wenn dabei die freie Verfügbarkeit des Quelltextes durch Erhalt der ursprünglichen Lizenz gewährleistet bleibt. · **Copyright** ist ein gesetzlicher Schutz im angloamerikanischen Rechtsraum, der Autoren, Künstler usw. vor unautorisierter Verbreitung ihrer Werke schützt. Im Unterschied zum deutschen Urheberrecht dient es hauptsächlich dazu, wirtschaftliche Investitionen zu schützen. Die Entscheidungs- und Verwertungsrechte können daher auch z. B. dem Verlag zustehen. · **Creative Commons** ist eine gemeinnützige Organisation, die verschiedene Lizenzen anbietet, mit denen Autoren allgemeine Nutzungsrechte an ihren Werken, wie z. B. Texten, Bildern, Musikstücken, einräumen können. Zu den einzelnen Abstufungen innerhalb der Lizenzen siehe auch S. 281. · **CRM** (Customer Relationship Management) bezeichnet die Verwaltung von

Kundenbeziehungen eines Unternehmens. · **Cross-Kompilieren** beschreibt den Vorgang, bei welchem der Quellcode eines Programmes zur Verwendung auf einer anderen Plattform als der, auf welcher die Kompilierung stattfindet, übersetzt wird. · **CUPS** (Common UNIX Printing System) ist der Standard-Druck-Server auf vielen Linux-Distributionen und in *Mac OS X*. CUPS verwendet das *Internet Printing Protocol (IPP)*, um Druckaufträge, Warteschlangen und Netzwerkdrucker zu verwalten. · **CVS** (Concurrent Versions System) ist ein Softwaresystem zur Verwaltung von Quelltexten und anderen Ressourcen in Form von Dateien. Als sog. Versionskontrollsystem ermöglicht CVS das Speichern und Verwalten verschiedener Versionen einer Datei. Dies ermöglicht das gemeinsame Arbeiten mehrerer Personen an einer Datei und die Wiederherstellung älterer Versionen (ähnlich der „Rückgängig-Funktion" in Office-Programmen). · **Cyberspace** ist eine Wortschöpfung des Autors William Gibson, die sich auf virtuelle künstliche Netze bezieht. Das Präfix *Cyber* wird oft für Begriffe verwendet, die mit Computern und Netzen verbunden werden. *Cyberspace* wird mittlerweile als Synonym für das Internet gebraucht. · **DHCP** (Dynamic Host Configuration Protocol) ermöglicht mit Hilfe eines entsprechenden Servers die dynamische Zuweisung von IP-Adressen wie auch weiterer Konfigurationsparameter an Computer eines Netzwerks. · **Digital Divide** (engl. für: Digitale Spaltung) bezeichnet die Abkopplung einzelner Länder oder Gebiete von der globalen Informationsgesellschaft durch technologischen Rückstand. · **DMZ** (Demilitarized Zone) ist eine Art neutrale Zone eines Sicherheitskonzeptes in bzw. zwischen Netzwerken. Sie befindet sich meist zwischen einem privaten Netzwerk (z. B. Intranet eines Unternehmens) und einem öffentlichen Netz (z. B. Internet). · **DNS** (Domain Name Service) ist ein Dienst, der zwischen IP-Adressen und leichter verständlichen Domainnamen übersetzt. So steht z. B. die Domain opensourcejahrbuch.de für die IP-Adresse 130.149.24.71. · **Embedded-Systeme** sind Softwaresysteme, die direkt in die Hardware eines Gerätes eingebaut sind, etwa im Automobil- oder Entertainment-Bereich. · **ERP** (Enterprise Resource Planning) bezeichnet die Aufgabe, die in einem Unternehmen vorhandenen Ressourcen, wie z. B. Kapital, Betriebsmittel oder Personal, möglichst effizient für den betrieblichen Ablauf einzuplanen. · **Fachverfahren** sind Verfahren oder Abläufe, die durch Software unterstützt werden. Diese meist in Organisationen (z. B. Behörden) eingesetzten Verfahren resultieren aus sehr speziellen Aufgaben (z. B. Verwaltung der Hundesteuer), die sich nicht mit Standardsoftware bearbeiten lassen. Deshalb wird die entsprechende Software meist in Eigenentwicklung oder von einem Dienstleister erstellt. · **Fat-Client** ist ein in ein Netzwerk eingebundener Rechner, der (im Gegensatz zum Thin-Client) Programme und Daten lokal vorhält. · **forking** (engl. *fork* = Gabel, Gabelung) bezeichnet bei der Softwareentwicklung das Auf- oder Abspalten in eigenständige und sich parallel weiterentwickelte Projekte. · **Freeware** ist Software, für die keine Lizenzgebühren entrichtet werden müssen. Die Nutzung ist für die ausführbare Version fast uneingeschränkt möglich und die Weiterverbreitung ist auch erlaubt. Der Quellcode ist jedoch meistens nicht verfügbar. · **GCC** (GNU Compiler Collection) ist eine umfangreiche Suite zur Übersetzung von Programmcode unterschiedlichster Sprachen in Maschinensprache. · **GIMP** (GNU Image Manipulation Program) ist ein unter der *GPL* lizenziertes, umfangreiches Bildbearbeitungsprogramm. · **GNOME** (GNU Network Object Model Environment) ist eine sehr beliebte

grafische Benutzeroberfläche für UNIX-Derivate, wie z. B. Linux. · **GNU** ist ein rekursives Akronym für „GNU's Not Unix". Dieses steht für ein von Richard Stallman ins Leben gerufenes Projekt (1983 wurde die Idee erstmals in einer Newsgroup gepostet) zur Entwicklung freier Software. · **GNU General Public License** siehe GPL. · **GPL** (GNU General Public License) ist eine Open-Source-Lizenz, die von Richard Stallman geschaffen wurde, um Software samt Quelltext effektiv und nachhaltig zu einer Art öffentlichem Gut zu machen. Für eine unter der *GPL* stehende Software gibt es keinerlei Restriktionen bezüglich Nutzung, Modifikation und Weitergabe außer den Pflichten des Copylefts: Bei Weitergabe müssen Lizenz und Quelltext mitgegeben oder anderweitig zugänglich gemacht werden, und Änderungen am Quelltext müssen wiederum unter der *GPL* stehen. · **Hacker** ist eine Person, die mit sportlichem Ehrgeiz die technischen Möglichkeiten von Hard- und Software zu ergründen und zu verwenden versucht. Daher kennt sich der Hacker gut mit Technik, insbesondere deren Stärken und Schwächen, aus. Hacken ist kreativer Umgang mit Technik. Teil der sog. Hacker-Philosophie ist, dass Informationen für jeden frei zugänglich sein sollen. Der Begriff „Hacker" ist urspr. wertneutral, wird aber von den Medien häufig i. Zshg. mit Computer-Kriminalität verwendet. · **Hebbsche Lernregel** ist eine nach Donald Hebb benannte Regel, die zuerst 1890 von William James formuliert wurde. Sie beschreibt den gegenseiten Einfluss benachbarter Neuronenverbände, die in zeiticher Nähe zueinander aktiv sind, auf die Stuktur ihrer Verbindungen. · **Howto** ist eine Anleitung oder ein *Tutorial* zu einem bestimmten Thema. · **HTTP** (Hypertext Transfer Protocol) ist ein Protokoll zum Übertragen von Text-, Grafik-, Sound- oder anderen Dateien im *World Wide Web*. · **HTTPS** (Hypertext Transfer Protocol over Secure Socket Layer) bedeutet die mit SSL verschlüsselte Übertragung von HTTP-Paketen. · **IEEE** (Institute of Electrical and Electronics Engineers) ist ein weltweiter Berufsverband von Ingenieuren aus den Bereichen Elektrotechnik und Informatik. Das Institut veranstaltet Fachtagungen, ist Herausgeber diverser Fachzeitschriften und bildet Gremien für die Normung von Techniken, Hardware und Software. · **Interoperabilität** beschreibt die Fähigkeit zur Zusammenarbeit von verschiedenen Komponenten (meist von unterschiedlichen Herstellern) eines heterogenen Gesamtsystems. Offene Standards begünstigen Interoperabilität. · **IRC** (Internet Relay Chat) ist ein Internetdienst, bei dem man sich in virtuellen Chaträumen, den sog. *channels*, mit anderen Personen zeitnah über Textnachrichten unterhalten kann. Man nennt dies auch „chatten". · **JSON** (JavaScript Object Notation) ist ein Computer-Format zum Datenaustausch zwischen Anwendungen. · **JTAG** (Joint Test Action Group) bezeichnet ein Verfahren zum Testen und Debuggen von elektronischer Hardware nach IEEE-Standard 1149.1. · **KDE** (K Desktop Environment) ist eine grafische Benutzer-Oberfläche v. a. für UNIX-Derivate, wie z. B. Linux. · **Kernel** ist der Kern eines Betriebssystems. Er koordiniert etwa die Verteilung der Hardware-Ressourcen an Applikationen und Zugriffe auf den Speicher. · **Knoppix** ist eine Linux-Distribution verbunden mit einer Live-CD oder -DVD, die ohne vorherige Installation auf einem Rechner von CD-ROM/DVD gestartet werden kann. Das von Klaus Knopper herausgegebene Knoppix ermöglicht ein Arbeiten mit vollwertiger grafischer Benutzeroberfläche inkl. Office-Programmen, ohne dass etwas an der Konfiguration des eingesetzten Rechners geändert werden muss. · **LAMP** steht als Akronym für eine Web-Plattform basierend auf den

OSS-Komponenten Linux, *Apache* und *MySQL*, in Verbindung mit *Perl*, *PHP* und *Python*. · **LDAP** (Lightweight Directory Access Protocol) ist ein Protokoll zum Zugriff auf einen Verzeichnisdienst, in dem Daten unterschiedlicher Art in einer festgelegten Struktur gespeichert werden können. · **Load Balancing** (Lastverteilung) ist eine Technik, um Berechnungen oder Anfragen auf mehrere Systeme zu verteilen. · **Maintainer** bezeichnet eine leitende Rolle in Open-Source-Projekten, die dazu dient, Entwicklungsaufgaben zu koordinieren. · **Migration** bedeutet die koordinierte Umstellung von Systemen, etwa die Einführung von Linux auf Arbeitsplatz-Rechnern. · **Multicast** oder Gruppenruf bezeichnet in der Telekommunikation eine Nachrichtenübertragung von einem Punkt zu einer Gruppe (auch Mehrpunktverbindung genannt). Der Vorteil von Multicast besteht darin, dass gleichzeitig Nachrichten an mehrere Teilnehmer oder an eine geschlossene Teilnehmergruppe übertragen werden können, ohne dass sich beim Sender die Bandbreite mit der Zahl der Empfänger multipliziert. · **NDA** (Non Disclosure Agreement) ist eine im kommerziellen Softwarebereich verbreitete vertragliche Vereinbarung über die Geheimhaltung von Geschäftsgeheimnissen, zum Beispiel Quelltexten und Schnittstellenspezifikationen. · **Nerd** ist die meist abwertend verwendete Bezeichnung für einen fachlich herausragenden Menschen mit geringen sozialen Fähigkeiten. Positiv und anerkennend ist der Begriff dagegen unter Gleichgesinnten (Nerds) und bei Eigenbezeichnungen gemeint. · **Newsgroups** sind elektronische Foren im Internet, in denen man Diskussionen führen und Meinungen austauschen kann. · **NGO** (Non-Governmental Organization) ist eine Nichtregierungsorganisation und meint prinzipiell jeden auf gewisse Dauer organisierten Zusammenschluss von Menschen, der nicht gewinnorientiert, nicht von staatlichen Stellen organisiert bzw. abhängig und auf freiwilliger Basis aktiv ist. · **OASIS** (Organization for the Advancement of Structured Information Standards) ist eine internationale Non-Profit-Organisation, die sich mit der Weiterentwicklung von u. a. Dokumenten-Standards wie *OpenDocument* beschäftigt. · **OpenLDAP** ist eine OSS-Implementierung des *LDAP*. · **OpenOffice.org** ist eine freie Office-Suite, die aus dem kommerziellen *StarOffice* von *Sun Microsystems* hervorgegangen ist. · **Open Source Initiative** siehe *OSI*. · **Open-Source-Software** bezeichnet nach Definition der *OSI* Programme, deren Lizenzen bestimmen, dass der Quelltext offen vorliegt sowie verändert werden darf und dass es erlaubt ist, die Software beliebig zu kopieren, zu verbreiten und zu nutzen. · **OSI** (*Open Source Initiative*) ist eine unabhängige Organisation, die Lizenzen nach der Open-Source-Definition zertifiziert. Dazu muss die Softwarelizenz die freie Verfügbarkeit, Weitergabe und Veränderbarkeit der Software fordern. · **OSS** siehe Open-Source-Software. · **OS X** ist eine Version von *Mac OS*, Apples Betriebssystem für Macintosh-Computer. · **PageRank** ist ein Algorithmus der untereinander verlinkte Dokumente anhand der Struktur ihrer Verlinkung bewertet. Er wurde von Larry Page und Sergey Brin entwickelt. · **Patch** ist eine Änderung an einer Software, die für das Hinzufügen einer Funktion oder die Fehlerbereinigung, wie etwa das Schließen einer Sicherheitslücke, verwendet wird. · **Peer–Review** ist die fachliche Überprüfung von Artikeln durch gleichgestellte Fachleute. · **Perl** ist eine sehr flexible Skriptsprache, die vom Administrationswerkzeug bis zum dynamischen Webseiten-Generator in nahezu allen computertechnischen Einsatzbereichen verwendet wird. · **PHP** ist eine Skriptsprache, die v. a. für die dynamische Erstellug von Webseiten verwendet

wird. · **Plug-ins** sind Zusatzmodule, die dazu dienen, Programme um bestimmte Funktionalitäten zu erweitern. · **proprietär** bezeichnet Software, Formate und Protokolle, die nicht „frei" sind, d. h. nicht (quell-)offen oder veränderbar. Nicht-standardisierte Entwicklungen werden ebenfalls als *proprietär* bezeichnet. · **Proxy** ist ein Dienst, der als „Mittler" fungiert. Er kann zwischen Rechnern, Netzwerken oder weiteren Diensten platziert werden und übernimmt dabei (je nach Auslegung) unterschiedliche Aufgaben. Beispielsweise kann ein Proxy eine Firewall (Nachrichten bzw. Pakete werden analysiert und ggf. ausgefiltert) oder einen Zwischenspeicher (*Cache* z. B. für Webseiten) darstellen. · **Python** ist eine objektorientierte Skriptsprache mit übersichtlicher Syntax. · **QPL** ist eine „Not-For-Profit" Softwarelizenz. Sie ist allgemein nicht kompatibel zur *GPL*, da man Veränderungen von QPL-lizenzierter Software nur in Form von Patches oder ähnlich deutlich gekennzeichnet verbreiten darf. Die *GPL* verbietet daher theoretisch die Verbindung (*linking*) von QPL-Software mit GPL-Software, manche Rechteinhaber wie die *Free Software Foundation* erteilen jedoch Sondergenehmigungen hierfür. · **RDF** (Resource Description Framework) ist eine Beschreibungssprache für Metainformationen *über* Webseiten. RDF-Anwendungsgebiete sind *Sematic Web* und RSS. · **Release** bezeichnet eine Softwareversion und den dazugehörigen Veröffentlichungstermin. · **REST** (Representational State Transfer) bezeichnet einen Softwarearchitekturstil für verteilte Hypermedia-Informationssysteme. · **Samba** ist OSS, mit der in einem Netzwerk Datei- und Druck-Dienste zur Verfügung gestellt werden, die sowohl von Linux- als auch von Windows-Clients genutzt werden können. · **Sendmail** ist eines der am weitesten verbreiteten Mail-Transfer-Programme im Internet. Es steht seit der Version 8.9 unter der *Sendmail License*, vormals stand es unter der BSD-Lizenz. · **Server** ist ein Rechner, der über ein Netzwerk Dienste anbietet, oder ein Programm, mit dem ein solcher Dienst bereitgestellt wird. · **Skript** ist genau wie ein Programm eine Reihe von Anweisungen für den Computer, die allerdings nicht zuvor kompiliert werden müssen. Bekannte Skriptsprachen sind *Perl*, *PHP*, *bash*, *Ruby*, *Python*, *Tcl* und *Applescript*. · **Smartcard** ist eine Chipkarte mit eingebautem Kryptoprozessor. Ein bekanntes Beispiel für eine Smartcard ist die Patienten-Chipkarte der Krankenkassen. · **Software-Stack** ist eine Menge von aufeinander aufbauenden Softwarekomponenten in einem größeren System. So bilden etwa Linux, *Apache*, *MySQL* und *Perl* den LAMP-Stack. · **SQL** (Structured Query Language) ist eine (teilweise genormte) strukturierte Abfragesprache zur Kommunikation mit Datenbanken. · **SSL** (Secure Socket Layer) ist ein Protokoll zum Verschlüsseln von Internet–Verbindungen. Es wird u. a. von Webbrowsern unterstützt und gewährleistet eine abgesicherte Übertragung von Nachrichten über das Internet. · **Stack** siehe Software-Stack. · **Structured Query Language** siehe SQL. · **TCO** (Total Cost of Ownership) umfasst die Gesamtbetriebskosten der Infrastruktur im laufenden Betrieb. Dabei werden nicht nur die budgetierten Kosten der IT-Abteilung (Hardware, Software, Personalkosten für Anschaffung und Maintenance), sondern auch die indirekt anfallenden Kosten bei den Endanwendern betrachtet. Üblich ist dabei ein Betrachtungszeitraum von einem Jahr. · **Thin-Client** ist ein Computer, der in einem Netzwerk als Endgerät (Terminal) dient. Benötigte Programme werden über das Netzwerk geladen, und Daten werden auf einem entfernten Speicher abgelegt (vgl. *Fat-Client*). · **Thread** (engl.: Faden, Strang) ist in der Informatik ein Programmteil, der zeitlich unabhängig von an-

deren Programmteilen abläuft. Im Zusammenhang mit E-Mails und Newsartikeln werden alle Beiträge, die sich auf eine Ausgangsnachricht bzw. einen Artikel beziehen, ebenfalls als Thread bezeichnet. · **Total Cost of Ownership** siehe TCO. · **Usability** bezeichnet die Benutzerfreundlichkeit einer Applikation. · **Usenet** war ein UUCP-basiertes Netzwerk in den 1980er und frühen 1990er Jahren für den regen Austausch in Diskussionsforen, sog. Newsgroups. Das UUCP-Verfahren (*Unix to Unix CoPy*) verlor durch den Boom des Internet stark an Bedeutung. · **Verzeichnisdienst** speichert Daten unterschiedlicher Art in einer festgelegten Struktur. Ein Verzeichnisdienst wie *OpenLDAP* stellt diese in einer IT-Infrastruktur zur Verfügung. · **VMS** Die Firma *Digital Equipment Corporation (DEC)* produzierte Maschinen zur elektronischen Datenverarbeitung, die mit dem Betriebssystem *VMS (Virtual Memory System)* ausgestattet waren. „Vater" dieses Betriebssystems war Dave Cutler, der später die Architektur von *MS Windows NT* gestaltete. · **Weblog** siehe Blog. · **Wiki** bezeichnet das Konzept von Websites, auf denen Besucher Texte nicht nur lesen, sondern auch unkompliziert bearbeiten können. Bekanntestes Beispiel ist die *Wikipedia*. · **XDMCP** (X Display Manager Control Protocol) ist ein Netzwerkprotokoll, dass der Kommunikation zwischen einem X-Terminal und einem X-Server dient. · **XML** (Extensible Markup Language) ist eine standardisierte Technologie zur Erstellung von Austauschsprachen und -formaten.

Stichwortverzeichnis

100-Dollar-Laptop ..... *siehe* OLPC

A

ABG-Recht 187
AbiWord 178
Access Linux Platform 47
ACL 259
Active Directory 259
Adobe 198
Africa i-Parliaments Action Plan 12
AGB-Recht 185–195
Alfresco 244
Altsysteme 169–181
Anwender
    ~betreuung 213
    ~emanzipation 7–17
    ~netzwerk 7–17
Apache 162, 164, 259
API 259
Apple 86
Archivierung 233–246
arXiv.org 227
ASCII 170
ASP 259
ASP.NET 205
ATOM 259
Auswärtiges Amt 212

B

B2C 259
Börries, Marco 28
Backend 259
Barebone 256, 259
Berliner Erklärung 259
Best Practice 259
BGB 186
Blackboard-System 259
Blog 259
bounded rationality 121
Branding 41–49
Bronner, Fall 109
Bug-Tracker 260
Bundesministerium für Bildung und Forschung 59
Bungeni 12
Business Intelligence 260

C

Card Sorting 260
CC 260
CeBIT 35
CentOS 72
Change Management 207–215
Client/Server-Architektur 171
Cluster 260
CMS 197–206, 260
Code
    ~-Review 260
Cofundus.org 51–59
Colocation 260
CommunesPlone.org 12
Community 27–40, 260
    ~ Council 31
    ~-Building 27–40
    ~-Mitglied 36
    verteilte ~ 69
Contenido 199
Content Management System ..... *siehe* CMS
Contribute 198
Contributor License Agreement 22
Converse 93
Copyleft ... 126, 176, 200, 226, 233–246, 260
Copyright 176, 260
Creative Commons 226
CRM 260
Cross-Kompilieren 261
CUPS 261
CVS 261
Cyberspace 261

D

Data-Warehouses 236
Debian 69
DEC 170

Deutsche Gesellschaft für Technische Zusammenarbeit 252
DHCP 261
Digital
~ Commons 223–232
~ Divide 261
~ Rights Management 233–246
Distributed Proofreaders 225
DMCA 229
DMZ 261
DNS 261
Document Management System 199
DotNetNuke 197–206
Dreamweaver 198
Drupal 197–206
DSpace 244
DZUG e.V. 13

E

E-Government 158
~-Initiative i2010 13
Edubuntu 253
EG-Vertrag 105
eID card 13
Eigentum
geistiges ~ 70, 108
Entwickler
~-Community 7–17
Entwicklung
~smodell 52–56
~sprojekt 249–257
Environment Information System 33
ERP 261
Essential-Facilities-Doktrin 108
Europäischer Gerichtshof (EuGH) 111
Europäisches Gericht erster Instanz (EuG) 105

F

Fabalabs Software GmbH 34
Fachverfahren 261
Fat-Client 261
Fedora Commons 243
Firefox ..... *siehe* Mozilla Firefox
Firmenbeteiligung 33
Flames 127
FLOSS 169–181
forking 10, 261
FOSDEM 141
Frauenquote 39
Free Software Foundation 128, 137
FreeSmartPhone.org 48
Freeware 261
Freie Software 65–75
FreiOSS 249

G

Gates, Bill 115
GCC 261
Genondatenbanken 227
GIMP 261
GNOME 261
GNU 223, 262
~ General Public License 185
~ General Public License .. 20, 66, 128, 137, 185–195, 200, 226, 262
~ Linux 164, 178, 207–215
Google 19, 131
~ Open Handset Alliance 48
~ Summer of Code 38
Grok 11

H

Hacker 262
Haftungsausschluss 185–195
Harvester-Software 233–246
Helfer 29
Hierachie 28
Howto 262
HTML 197
HTTP 262
HTTPS 262

I

IAO 156
IBM 33, 170
Iceweasel 71
IEEE 262
IMS Health 114
Intel 85, 171
Interoperabilität 107, 175, 262
IRC 32, 262
IT
~-Unternehmen 155–168

J

JBOSS 162, 164
Joomla 197–206
JSON 262
JTAG 262

K

KDE 34, 128, 262

Kernel 262
Kiwix 253
Knoppix 262
kommunikative Netzwerke 135–147
Kompatibilität 174

L
LAMP 262
Landgericht München 185
LDAP 263
LibriVox 225
LiMo-Foundation 47
LiMux 34
Linux4Afrika 249–257
LinuxBIOS 178
LinuxTag 252
LiPSForum 48
Lizenz
~bedingungen 185–195
~vertrag 185
BSD-~ 260
Load Balancing 263
Lokalisierung 29
LTSP 250
Lutece d'Or 13

M
Mac OS X 27, 263
Maemo 41–49
Magill 112
Mailingliste 31
Maintainer 263
Mambo 199
market of lemons 172
Marktzugang 111
Messung 169–181
Micropledge 58
Microsoft 85, 105, 131, 139, 171
~ Office 207, 213
Der Fall ~ 105–115
Migration 207–215, 263
Mobilkommunikation 41–49
Monopol 65–75, 105–115
geistiges ~ 223–232
Moodle 253
Motivation 38, 91–101, 135–147
Mozilla 156
~ Firefox 51, 71, 93, 161
~ Foundation 71, 93
~ Thundebird 156
~ Thunderbird 161, 253
Multicast 263
MySQL 19–26, 70, 162, 164

N
Napster 126
NDA 263
Neo 1973 41–49
Nerd 263
Newsfeed 198
Newsgroup 263
NGO 263
Novell 34, 139
Nullsummen-Prinzip 79
Nutzungsrecht 190

O
O'Reilly, Tim 124
OASIS 263
Offene
~ Architekturen 169
~ Formate 169–181
~ Standards 169–181
Offenheit 169–181
Office 27–40
~-Suite 263
OLPC 178
Online-Community 51–59
OOoCamp 36
Open
~ Access 169, 223–232
~ Content 223–232
~ Data 223–232
~ Innovation 77–88
Open Source 1–273
~ Initiative ..... *siehe* OSI
~-Geschäftsmodell 77–88
~-Lizenz 200
~-Marketing 91–101
~-Philosophie 106
~-Projekt 27–40
~-Software 169–181, 263
OpenDocument 242
OpenEmbedded 41–49
OpenID 56
OpenLDAP 263
OpenMoko 41–49
OpenOffice.org 16, 27–40, 51, 161, 213, 253, 263
~ Deutschland e. V. 35
~-Konferenz 32
OpenXML 242
Opt-In Copyright 240
Oracle 85

Organisation 29
OSI 66, 263

P
Partizipation 7–17, 19–26
Patch 263
Patente 81–82, 176
Patentschutz 110
Peer-Review 263
Perl 263
PHP 21, 164, 263
phpMyAdmin 22
Plone 9, 15, 197–206
~ Improvement Proposal 9
PloneGov 7–17
Plug-in 264
Pootle 33
Portal 198
Portlet 198
Project Gutenberg 225
Projekt
~management 208
Accepted-~ 30
Incubator-~ 30
Native-Lang-~ 29, 30
PrOOo-Box 29
proprietär 264
Proxy 264
Public Domain 233–246
Public Library of Science 227
Python 9, 15, 21, 206, 264
~ Enhancement Proposal 9

Q
Qemu 41–49
QPL 264

R
Raymond, Eric S. 143
RDF 264
Recht
~emanagement 199
Europäisches ~ 105–115
Kartell~ 106
Urheber~ 106
Records Management 233–246
Red Flag 34
Red Hat 34, 86, 139
~ Enterprise Linux 72
Release 264
REST 264
RHEL ..... *siehe* Red Hat Enterprise Linux
Ruby 21

S
Samba 264
SAP 85
Saure-Gurken-Problem *siehe* market of lemons
Schadensersatz 193
Science Commons 226
Selbstregulierung 30
Sendmail 264
Server 264
Simon, Herbert 121
Skalierbarkeit 199
Skript 264
Skype 189, 194
Slashdot.org 127
Smartcard 264
SMC Networks 193
SOAP 159
Softwarebörse 51–59
Softwareentwicklung
kollaborative ~ 51–59
SoftWiki 59
Sponsoring 173
Sprint 11–12
SQL 264
SSL 264
Stallman, Richard M. 128, 137, 262
Star Division 28
StarOffice 28
Strategie 77–88
Studie 155–168
Sun Microsystems 16, 28, 108, 170, 263
SuSE 34

T
Tagging 233–246
TCO 168, 264, 265
Teilen 223–232
Terminalserver 249–257
Thin-Client 264
Thread 264
Total Cost of Ownership ..... *siehe* TCO
Tragedy of the Commons 225
Typo3 ..... 164, 197–206

U
Ubiquitous Computing 41–49
UdalPlone 12
United Nations Department of Economic and Social Affairs (UNDESA) ... 12
UNIX 170

Unterlassung 193
Urheber 186
    ~schaft 192
    ~vermutung 192
Usability 265
Usenet 265

V
Verfügbarkeit 175
Vergemeinschaftung 135–147
Vertragsschluss 186
Verwaltung 155–168, 212–214
Verzeichnisdienst 265
VMS 265

W
Web Content Management 198
Web Service 198
Webbrowser 171
Weblog ..... *siehe* Blog
Welte, Harald 193
Wettbewerb 106
Wiki 265
Wikipedia 19
Wirtschaftliche Auswirkungen 155–168
Wissen
    akkumuliertes ~ 8
Workflow 198

X
x86 171
Ximian 34
XML 159, 204, 265
XO ..... *siehe* OLPC

Y
Young, Bob 124

Z
Zea Partners 13
Zope 7–17
Zwangslizenz 114

Mitwirkende

Kaj Arnö ist nicht nur Geschäftsführer der *MySQL GmbH*, sondern daneben *Vice President Community Relations* von *MySQL AB*. Er verbindet in dieser Eigenschaft die internen mit den externen MySQL-Entwicklern. Er ist davon überzeugt, dass es eine tiefe Verbindung zwischen seinen Lieblingsthemen gibt. Zu diesen zählen das Leiten der Teams aus verschiedenen Kulturen und Zeitzonen sowie das Singen von Schnappsliedern – eine Gewohnheit aus seiner Heimat Finnland, die er mit wechselhaftem Erfolg als „zugeroaster" Münchner weltweit verbreitet. So hat er am 16. Januar 2008 anlässlich der bevorstehenden Übernahme durch *Sun Microsystems* vor dem gesamten MySQL- und Teilen des Sun-Personals das Lied *Helan går* angestimmt.

Daniel Auener studiert Informatik an der Technischen Universität Berlin. Sein Studienschwerpunkt ist „Informatik und Gesellschaft". Er übernahm in diesem Jahr die Leitung des Projekts.

Sören Auer (Dr.) leitet die Arbeitsgruppe *Agile Knowledge Engineering and Semantic Web* (AKSW) an der Abteilung Betriebliche Informationssysteme der Universität Leipzig. Er ist Gründer von verschiedenen Forschungsprojekten, wie z. B. der Wikipedia-Semantifizierung *DBpedia*, Autor von mehr als 50 wissenschaftlichen Publikationen und Mitglied des Advisory–Boards der *Open Knowledge Foundation*.

Matthias Bärwolff ist wissenschaftlicher Mitarbeiter von Prof. Lutterbeck am Fachgebiet Informatik und Gesellschaft der Technischen Universität Berlin.

Leonie Bock ist wissenschaftliche Mitarbeiterin am Lehrstuhl von Prof. Dr. Dr. J. Ensthaler an der Technischen Universität Berlin. Ihr Forschungsschwerpunkt liegt auf den Gebieten Urheberrecht, Gewerblicher Rechtsschutz und Kartellrecht. Außerdem hat sie bei der Beratung öffentlicher und privater Unternehmen im Energierecht zu Fragen des Energiemanagements und Vergaberechts sowie der Entflechtung mitgearbeitet.

Nicole Bräuer hat auch in diesem Jahr wieder das Lektorat für das Buch übernommen. Dabei hat sie unter äußerster Termindichte jeden noch so kleinen Fehler in den Texten aufgespürt.

Richard Bretzger studiert Soziologie technikwissenschaftlicher Richtung mit den Nebenfächern Informatik und Sozialpsychologie an der Technischen Universität Berlin und hat für dieses Jahrbuch Autorinnen und Autoren für die soziologische Perspektive auf Open Source akquiriert und betreut.

Thorsten Busch hat an den Universitäten Oldenburg und St. Gallen Politik- und Wirtschaftswissenschaften studiert. Seit September 2007 ist er Doktorand und wissenschaftlicher Assistent von Prof. Dr. Peter Ulrich am Institut für Wirtschaftsethik der Universität St. Gallen.

Mark Cassell (Prof. Dr.) ist *Associate Professor* der Politikwissenschaft an der Kent State University (Ohio, USA). Sein Buch „How Governments Privatize: The Politics of Divestment in the United States and Germany" (Georgetown University Press, 2002) gewann den von der *International Political Science Association* ausgeschriebenen *Charles H. Levine Award* als bestes Buch im Bereich öffentliche Verwaltung und öffentliche Verfahrensweisen. Seine

Forschungsinteressen schließen die Reform des öffentlichen Sektors, Privatisierungen, Banken- und Finanzwesen und die Adaption neuer Technologien im öffentlichen Sektor ein.

Matthias Choules studiert Informatik an der Technischen Universität Berlin mit dem Schwerpunkt „Informatik und Gesellschaft". Er war verantwortlich für die Präsentation des Open Source Jahrbuchs auf der *CeBIT* und dem *LinuxTag*.

Bob Dannehl studiert Informatik an der Technischen Universität Berlin mit den Schwerpunkten „Datenbanken und Informationssysteme" sowie „Informatik und Gesellschaft". Neben der Verantwortlichkeit für den Bereich Sponsoring, fokussierte er sich bei der Autorenakquise vor allem auf große Open-Source-Projekte.

Oliver Diedrich (Dr.) arbeitet beim Heise-Verlag und ist dort für die Berichterstattung zu Linux in *c't* sowie die Open-Source-Website *heise open* verantwortlich. Er schreibt seit zehn Jahren über freie Software – und verwendet sie täglich selbst.

Robert A. Gehring studierte Elektrotechnik, Informatik und Philosophie an der Technischen Hochschule Ilmenau und an der Technischen Universität Berlin. Dort promoviert er derzeit zu Fragen der Softwareökonomie und des Softwarerechts. Seit Ende 2005 ist er Mitglied der Redaktion des Urheberrechtsportals *iRights.info*, das 2006 mit dem Grimme-Online-Preis ausgezeichnet wurde. Als freiberuflicher Consultant berät er namhafte deutsche Unternehmen in IT-Fragen.

Mathias Eberle studiert Informatik an der Technischen Universität Berlin mit den Schwerpunkten „Betriebs- und Kommunikationssysteme" und „Informatik und Gesellschaft". Daneben ist er Mitinhaber des Musikverlags *Edition Punctum Saliens*. Für das aktuelle Jahrbuch war er für den Textsatz verantwortlich.

Beate Groschupf studierte an der Technischen Universität Berlin Erziehungswissenschaften und Pädagogik. Seit 2003 übernimmt sie die Projektorganisation für die Konferenz *Wizards of OS*. Als Inhaberin des Beratungsunternehmens *CHANGE Management für Veränderung* arbeitet sie zudem als Beraterin bei der Einführung von Open Source in Wirtschaft und Verwaltung.

Jochen Günther ist seit Oktober 2001 wissenschaftlicher Mitarbeiter am Fraunhofer-Institut für Arbeitswirtschaft und Organisation (IAO) im Marktstrategieteam *Business Performance Management*. Im Rahmen von Beratungs- und Forschungsprojekten beschäftigt er sich dort mit dem Einsatz von Informationstechnologie in Geschäftsprozessen.

Jan Ulrich Hasecke studierte Theater-, Film- und Fernsehwissenschaft in Köln und arbeitet als PR-Berater, Publizist und Autor. Er ist Mitglied der *Plone Foundation* und Vorstandsmitglied im *DZUG e.V.* (Deutschsprachige Zope User Group), den er zusammen mit anderen Anwendern des Webapplikationsservers Zope im Jahre 2004 gegründet hat.

Tobias Hauser ist Autor, Trainer und Berater mit dem Spezialgebiet Webtechnologien. Als Teilhaber der Agentur *Arrabiata Solutions GmbH* implementiert er Weblösungen, führt Code- und Usability-Audits durch und berät Kunden im In- und Ausland bei ihrer Webstrategie.

Janine Heinrich studiert Diplom-Übersetzen (Englisch/Französisch) an der Humboldt-Universität zu Berlin. Für dieses Buch übersetzte sie den Artikel von Glyn Moody.

Andrea Hemetsberger (Prof. Dr. rer. soc. oec.) forscht und lehrt im Bereich Marketing am *Institut für Strategisches Management, Marketing und Tourismus* an der Leopold-Franzens–Universität Innsbruck. Sie hat ihre Habilitation der F/OSS-Bewegung gewidmet und deren

Motivationen, deren System des offenen Austauschs, ihre widerständische Kultur und auch die Kollaboration und kollektive Wissenskreation in konkreten Open-Source-Projekten analysiert.

Andreas Hepp (Prof. Dr.) ist Professor für Kommunikationswissenschaft und Sprecher des Instituts für Medien, Kommunikation und Information am Fachbereich Kulturwissenschaften der Universität Bremen. Er lehrt im Bachelorstudiengang Kulturwissenschaft sowie im Masterstudiengang Medienkultur. Seine Schwerpunkte in Forschung und Lehre sind unter anderem Medien- und Kommunikationstheorie, Mediensoziologie, inter- bzw. transkulturelle Kommunikation sowie Medienrezeption/-aneignung und Diskursanalyse.

Georg Hüttenegger (Dr.) arbeitet als Software-Architekt für die Raiffeisen Software Solution und Service GmbH in Wien. 2004 promovierte er an der TU Wien über das Thema „Knowledge Management & Information Technology". Im Jahr 2006 veröffentlichte er das Buch „Open Source Knowledge Management" im Springer Verlag und arbeitet seit September 2006 als nebenberuflicher Lektor an der Fachhochschule Technikum Wien an den Themen Wissensmanagement und IT-Markt.

Christoph Jeggle war lange Jahre als Softwareentwickler und Entwicklungsleiter unter anderem bei einem Hersteller elektronischer Archivsoftware tätig und ist heute Projektleiter und Seniorberater bei *PROJECT CONSULT Unternehmensberatung Dr. Ulrich Kampffmeyer GmbH*, Hamburg.

Isabella Kahler studiert Interkulturelle Fachkommunikation (Übersetzen und Dolmetschen) mit den Sprachen Englisch und Russisch an der Humboldt-Universität zu Berlin.

Ulrich Kampffmeyer (Dr.) ist Gründer, Geschäftsführer und Chefberater der PROJECT CONSULT Unternehmensberatung Dr. Ulrich Kampffmeyer GmbH, Hamburg, eine produkt- und herstellerunabhängige Beratungsgesellschaft für Informationsmanagement (IM). Er berät Kunden aller Branchen im In- und Ausland bei Strategie, Konzeption, Auswahl, Einführung, Ausbau, Migration und Dokumentation von IM-Lösungen.

Christoph Koenig promoviert am Graduiertenkolleg „Qualitätsverbesserung im E-Learning durch rückgekoppelte Prozesse" an der TU Darmstadt über Lern- und Bildungsprozesse in kollaborativen Online-Communitys mit Schwerpunkt auf Open Source.

Sascha Langner ist wissenschaftlicher Mitarbeiter am Institut für Marketing und Management von Prof. Wiedmann an der Leibniz Universität Hannover. Er ist langjähriger Fachautor und Herausgeber des *eZines marke-X*, das sich an kleine und mittelständische Unternehmen richtet und innovative sowie kosteneffiziente Marketingstrategien behandelt.

Michael Lauer ist IT-Freelancer, Buchautor und Free-Software-Enthusiast, spezialisiert auf mobile verteilte Systeme, Embedded Linux und GUI-Toolkits. Er ist seit Beginn des OpenMoko-Projekts maßgeblich an der Entwicklung der Plattform beteiligt.

Bernd Lutterbeck (Prof. Dr. iur.) studierte Rechtswissenschaften und Betriebswirtschaftslehre in Kiel und Tübingen. Seit 1984 ist er Professor für Wirtschaftsinformatik an der Technischen Universität Berlin mit den Schwerpunkten Informatik und Gesellschaft, Datenschutz- und Informationsrecht sowie Verwaltungsinformatik. Seine Arbeitsschwerpunkte sind E-Government, Theorie und Praxis geistigen Eigentums, Open Source und *European Governance*.

Hans-Peter Merkel ist seit vielen Jahren in der Open-Source-Community aktiv. Sein Schwerpunkt liegt im Bereich Daten-Forensik und IT-Security. Er engagiert sich in der technischen

Entwicklungszusammenarbeit und ist Mitgründer und Vorsitzender von FreiOSS.net und Linux4Afrika.

Glyn Moody schreibt seit über einem Vierteljahrhundert über Computer-Technologie. Im Jahr 1997 verfasste er den ersten öffentlich beachteten Beitrag über GNU/Linux und freie Software, welcher im *Wired Magazine* (http://www.wired.com/wired/archive/5.08/linux_pr.html) veröffentlicht wurde. Sein Buch „Rebel Code: Inside Linux and the Open Source Revolution" wurde im Jahr 2001 vom *Penguin-Verlag* veröffentlicht und ist eine grundlegende, historische Betrachtung zu freier Software und Linux. Sein Blog, *Opendotdotdot*, befasst sich hauptsächlich mit Open Source, Open Content und Offenheit im Allgemeinen (http://opendotdotdot.blogspot.com/).

Lars Pankalla ist wissenschaftlicher Mitarbeiter am Institut für Marketing und Management von Prof. Wiedmann an der Leibniz Universität Hannover. Seine Arbeits- und Forschungsschwerpunkte liegen im Bereich des *Web 2.0* sowie des Innovations- und Technologiemanagements.

Henriette Picot (RA Dr.) ist als Rechtsanwältin in der Praxisgruppe IT bei *Bird & Bird* tätig und dort insbesondere auf urheberrechtliche Fragen spezialisiert. Sie berät Anbieter und Nutzer von (proprietärer und freier) Software und IT-Dienstleistungen in Fragen nationaler und internationaler Vertragsgestaltung (insbesondere zu Lizenzierung, Vertrieb und IT-Projekten), zu regulatorischen Fragen insbesondere im Bereich von Datenschutz und Datensicherheit sowie zu Fragen des allgemeinen Handels- und Gesellschaftsrechts. Sie studierte Rechtswissenschaften an den Universitäten Freiburg, Sevilla und Dresden.

Andi Pietsch ist Projektleiter für CMS- und Entwicklungsprojekte bei *Arrabiata Solutions GmbH*. Seine Aufgaben reichen von der Systementscheidung über die Projektplanung bis zum Projektcontrolling. Er berät und betreut CMS-Projekte für Startups, etablierte Mittelständler und Markenartikler.

Jacqueline Rahemipour ist seit Januar 2005 die Projektleiterin des deutschsprachigen Projekts von *OpenOffice.org* (http://de.openoffice.org). Außerdem ist sie Gründungsmitglied des Vereins *OpenOffice.org Deutschland e.V.* (http://www.ooodev.org). Schwerpunkte ihrer Arbeit sind Qualitätssicherung, Lokalisierung/Übersetzung und Marketing. Sie hat sich mit ihrer Dortmunder Firma, der *natural computing GmbH*, auf Linux am Arbeitsplatz spezialisiert und bietet darüber hinaus Beratung, Schulungen und Migrationsunterstützung für *OpenOffice.org* an. Im Oktober 2007 ist ihr Buch „Textverarbeitung mit OpenOffice.org 2.3 Writer" im Galileo-Press–Verlag erschienen.

Roman Rauch studiert Informatik an der Technischen Universität Berlin mit dem Schwerpunkt „Informatik und Gesellschaft".

Sebastian Röder studiert Soziologie technikwissenschaftlicher Richtung an der Technischen Universität Berlin. Er hat in diesem Jahr unter anderem den Versand des Open Source Jahrbuchs übernommen.

Marc Schachtel studiert Informatik an der Technischen Universität Berlin mit den Schwerpunkten „Computer Graphics" und „Informatik und Gesellschaft". Für das aktuelle Jahrbuch übernahm er die grafische Gestaltung und die Betreuung des Drucks.

Matthias Schenk studiert Informatik an der Technischen Universität Berlin mit den Schwerpunkten „Datenbanken und Informationssysteme" sowie „Informatik und Gesellschaft".

Malte Schmidt-Tychsen studiert Betriebswirtschaftslehre an der Technischen Universität Berlin. Seine Schwerpunkte sind „Rechnungslegung" sowie „Finanzierung und Investition".

Nadia Schüler ist IFK-Studentin (Dolmetschen und Übersetzen/Englisch und Französisch) an der Humboldt-Universität zu Berlin und hat für dieses Jahrbuch den Text von Joel West übersetzt.

Jan Suhr arbeitet derzeit als Unternehmensberater und studierte zuvor Informatik an der TU Berlin und an der Universität Hamburg. In seiner Abschlussarbeit beschäftigte er sich mit der Messung von Offenheit bei IT-Artefakten. Darüber hinaus war Jan Suhr für die TU Berlin in mehreren Entwicklungsprojekten in Afghanistan tätig.

Daniel Tepe ist Diplom-Pädagoge und wissenschaftlicher Mitarbeiter im Bereich E-Learning an der Universität Bremen. Als Doktorand am Institut für Medien, Kommunikation und Information (IMKI) beschäftigt er sich mit der Domestizierung von Bildungstechnologien am Beispiel von Lernplattformen. Daneben gehört er zu den Initiatoren des Internetportals für Szenenforschung *jugendszenen.com*, welches er seit 2001 hauptverantwortlich betreut.

Udo Thiedeke (Dr. phil. habil.) lehrt und forscht als Privatdozent am Institut für Soziologie der Johannes-Gutenberg-Universität in Mainz. Seine Arbeitsschwerpunkte sind Allgemeine Soziologie unter Berücksichtigung der Soziologie der Medien und der virtualisierten Vergesellschaftung, Soziologie der Politik und Soziologie der Bildung.

Joel West (Prof. Dr.) ist *Associate Professor* für *Innovation & Entrepreneurship* am Fachbereich für Organisation und Management im *College of Business* an der San José State University. San José ist die selbst ernannte „Hauptstadt des Silicon Valley". Wests Forschung konzentriert sich auf die Langzeitauswirkungen technologischen Wandels auf die industrielle Struktur, insbesondere Standards, Open-Source-Software und Open Innovation. Er ist zusammen mit Henry Chesbrough and Wim Vanhaverbeke Herausgeber des Buchs „Open Innovation: Researching a New Paradigm" (Oxford University Press, 2006) und arbeitet momentan an einem Buch über *Qualcomm* und die US-amerikanische Telekommunikationsindustrie.

Klaus-Peter Wiedmann (Prof. Dr.) ist Inhaber und Leiter des Instituts für Marketing und Management an der Leibniz Universität Hannover. Er ist gleichzeitig Visiting Professor am Henley Management College (UK) und Mitglied des Regional Committee of the American Chamber of Commerce (Frankfurt a. M./Hannover) sowie des Editorial Board der Zeitschrift „Corporate Reputation Review". Für das internationale „Reputation Institute" ist Professor Wiedmann als RI Country Director tätig.

Natascha Zorn ist Mitinhaberin von CHANGE Management für Veränderung, einem Beratungsunternehmen, das sich auf Veränderungsprozesse bei IT-Migrationen spezialisiert hat. CHANGE unterstützt seit 2005 Wirtschaft und Verwaltung insbesondere bei der Bewältigung von Anwenderwiderständen und -ängsten bei Linux-Migrationen.

Das Open-Source-Jahrbuch-Projekt

Wo am Anfang eine Vision war, stehen heute fünf Ausgaben des Open Source Jahrbuchs, welches inzwischen zu einer Standardreferenz im deutschsprachigen Raum gewachsen ist.
Herausgegeben wird das Jahrbuch am Fachbereich Informatik & Gesellschaft der Technischen Universität Berlin. Ein jährlich wechselndes, studentisches Team übernimmt dabei sämtliche anfallenden redaktionellen Aufgaben, von der Auswahl und fachlichen Betreuung der Autoren, Erarbeitung von Themenfeldern und Buchstruktur, über sämtliche Arbeiten der Druckvorstufe bis hin zu Marketing, Vertrieb und Geschäftsentwicklung.
Seit September 2006 stehen die Arbeiten des Projektes und der dem Dach des *Fördervereins Open Source Jahrbuch e. V.* Falls Sie Interesse haben, sich unterstützend an dem Projekt zu beteiligen oder als Autor mitzuwirken, finden Sie weitere Informationen auf unserer Website unter http://www.opensourcejahrbuch.de.

Lizenzen

Creative Commons

- Sie dürfen den Inhalt vervielfältigen, verbreiten und öffentlich aufführen.
- Nach Festlegung des Rechtsinhabers können die folgenden Bedingungen gelten:

   *Namensnennung* (BY) Sie müssen den Namen des Autors/Rechtsinhabers nennen.

   *Keine kommerzielle Nutzung* (NC) Dieser Inhalt darf nicht für kommerzielle Zwecke verwendet werden.

   *Weitergabe unter gleichen Bedingungen* (SA) Wenn Sie diesen Inhalt bearbeiten oder in anderer Weise umgestalten, verändern oder als Grundlage für einen anderen Inhalt verwenden, dann dürfen Sie den neu entstandenen Inhalt nur unter Verwendung identischer Lizenzbedingungen weitergeben.

   *Keine Bearbeitung* (ND) Der Inhalt darf nicht bearbeitet oder in anderer Weise verändert werden.

- Im Falle einer Verbreitung müssen Sie anderen die Lizenzbedingungen, unter die dieser Inhalt fällt, mitteilen.
- Jede dieser Bedingungen kann nach schriftlicher Einwilligung des Rechtsinhabers aufgehoben werden.
- Die gesetzlichen Schranken des Urheberrechts bleiben hiervon unberührt.
- Nachfolgend die CC-Lizenzen der Version 2.5:

  Attribution (BY)
  http://creativecommons.org/licenses/by/2.5/

  Attribution-NoDerivs (BY-ND)
  http://creativecommons.org/licenses/by-nd/2.5/

  Attribution-NonCommercial-NoDerivs (BY-NC-ND)
  http://creativecommons.org/licenses/by-nc-nd/2.5/

  Attribution-NonCommercial (BY-NC)
  http://creativecommons.org/licenses/by-nc/2.5/

  Attribution-NonCommercial-ShareAlike (BY-NC-SA)
  http://creativecommons.org/licenses/by-nc-sa/2.5/

  Attribution-ShareAlike (BY-SA)
  http://creativecommons.org/licenses/by-sa/2.5/

  Eine Übersicht befindet sich auch unter http://creativecommons.org/licenses/.

Relevanz kennt kein Verfallsdatum

Das Phänomen „Open Source" fasziniert heute mehr denn je. Dabei haben die meisten Artikel der Jahrbuch-Reihe ihre Gültigkeit bis heute behalten. Seitdem 2004 erstmals das umfassende Kompendium mit grundlegenden Erkenntnissen rund um das Prinzip der offenen Quelle zusammengetragen wurde, erfreut sich die Mischung von Beiträgen aus Wirtschaft, Verwaltung, Wissenschaft und Praxis großer Beliebtheit:

«Neben der Bündelung des fachlichen Hintergrundwissens zeigt das Buch ausgehend von Fallbeispielen auch die Potenziale in großen Unternehmen und Behörden auf. . . . Softwareentwicklung zeigt plötzlich zahlreiche rechtliche, ökonomische und sogar gesellschaftliche Facetten.»

Maik Schmidt, c't – magazin für computertechnik, Buchkritik Open Source Jahrbuch 2005

«Der Band, an dem 39 Autoren mitgewirkt haben, erhebt zu Recht den Anspruch darauf, ein aktuelles Nachschlagewerk für Wirtschaft, Verwaltung und Wissenschaft zu sein.»

Rainer Hill, Kommune21, Buchvorstellung des Open Source Jahrbuch 2006

«Beeindruckend am Gesamtwerk erscheint auch bei dieser Ausgabe wieder die enorme Themenvielfalt der Artikel. . . . Eine rundum gelungene Artikel-Zusammenstellung mit interessanten Einblicken in die Zukunft von Open Source.»

T3N Magazin für Open Source und Typo 3, Ausgabe 7, zum Open Source Jahrbuch 2007

Open Source Jahrbuch 2004, 2005, 2006 und 2007

Die komplette Reihe ist erhältlich unter http://opensourcejahrbuch.de und via Lehmanns Fachbuchhandlung.